# A NETWORK-ON-CHIP ROUTER FOR LOW-LATENCY AND HIGH-THROUGHPUT DIMENSION-ORDER ROUTING

A Thesis

by

SHALIMAR RASHEED

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

| | |
|---|---|
| Chair of Committee, | Jiang Hu |
| Co-Chair of Committee, | Paul V. Gratz |
| Committee Member, | Duncan M. Walker |
| Head of Department, | Chanan Singh |

May 2014

Major Subject: Computer Engineering

# ABSTRACT

Networks-on-Chip (NoCs) offer a scalable means of on-chip communication for future many-core chips. As NoC size increases with core count in future many-core chips, NoC router efficiency is critical to achieving performance scaling. This thesis explores router microarchitectures which leverage traffic pattern biases and imbalances to reduce latency and improve network throughput. It introduces STORM – Simple Traffic-Optimized Router Microacrchitecture – a new, low-latency, fair, high-throughput NoC router design, customized for the traffic seen in a two-dimensional mesh network employing dimension-order routing. Compared to a baseline NoC router with equivalent buffer resources, STORM offers single cycle operation *and* reduced cycle time (up to 17% less than the baseline on 45nmCMOS), with less area and power consumption when synthesized at the same clock frequency as the baseline. This design yields a higher overall network saturation throughput (up to 14.6% higher than the baseline) in an 8x8 2D mesh network for uniform random traffic. STORM also reduces packet latencies under realistic workloads by 41% on average.

# DEDICATION

To my parents, to whom I owe all I am and all I have.

ACKNOWLEDGEMENTS

I have several people to thank for assisting me throughout my work on this thesis. I am most grateful to Professors Jiang Hu, Paul Gratz and Srinivas Shakkottai, my advisors for this project.

The guidance I have received from Professor Gratz and Professor Hu has been nothing short of amazing. They have been very enthusiastic about my research, giving me novel ideas and pointing me to valuable resources. Professor Gratz has provided invaluable insights from his vast practical experience on Networks-on-Chip and has lent clear direction to my work. Professor Hu has been instrumental in keeping me on track with timely encouragement and support. Professor Shakkottai's knowledge of macro-networks has helped me understand Networks-on-Chip better, and I thank him for his guidance.

I also express my gratitude to Professor Duncan M. Walker, from the department of Computer Science, for agreeing to be on my advisory committee and for his valuable comments.

My fellow students deserve plenty of credit for supporting me, in various ways. Ruixiao Ni, Jae-Yeon Won, Hyungjun Kim, Mark Browning, Aravind Natarajan, Rohit Kumar, Kaustubh Gupta, Anil Alagiya, Ashish Singla, and Zheng Xu. All these wonderful people have my sincere gratitude, and it has been a privilege to work with them and benefit from their help.

My thanks are due to the faculty and staff of Electrical and Computer Engineering at Texas A&M for all their wonderful help over the past two years.

I also extend my sincere thanks to my friends from the College of Engineering, Guindy, (CEG) Chennai, India, Class of 2012, and to my undergraduate professors. Special thanks to Dr. K. Gunaseelan, Dr. C. M. Sujatha, Dr. Kiran Kuchi, Dhivagar Baskaran, Dr. Anand Kannan, VEPL, India. I am indebted to them for guiding and encouraging me to enroll in a graduate program. My senior students from CEG who helped me with my general questions on graduate life have my deep gratitude.

My friends from middle and high school – Jose, Kannan, Karthik, Dinesh J, Dinesh M, Vikram, Logesh, Velmurugan, Deepak, Anantharaghavan, and others – also have my gratitude, for encouraging me to follow my dreams and overcome my inhibitions, and for all the fun we had together.

I would be committing cardinal sin by not acknowledging the priceless support of my friends in College Station and elsewhere in the United States: Sindhu Alagesan, Nelson Antony, Arun Balaji, Sheffali Chugh, Raveesh Chugh, Sivasankar Dhanapal, Chris Hoefler, Arun Ilango, Jayanth Jayaraman, Karthikeyan Kalidasan, Bita Kash, Fred Kash, Ganapathy Kasi, Rajesh Kumar, Jagadish Kumaran, Deepak Muthukrishnan, Taraz Nosrat, Ziba Nosrat, Mack Nosrat, Kia Parsi, Catherine Parsi, Alagappan Ponnalagu, Karthik Prabhu, Manoj Prasad, Preshmitha Rajagopal, Kousik Ramasamy, Parthiban Ravikumar, Gowtham Ravisankar, Ashwini Ravindran, Morgan Suhm, Marissa Suhm, Grant Suhm, Afsaneh Yazdani, Baman Yazdani. These wonderful people

have helped me in myriad ways, too many to recount here. I have the deepest gratitude for all of them.

Manoj Baskaran (TAMU, 2012) and Prithivi Tamilarasan (TAMU, 2013) were and continue to be my go-to people for academic and general guidance. Manoj's astute academic and career advice has helped me use the last two years of my life in a very productive way. Prithivi has been a guiding light throughout my graduate life, helping me out with plenty of academic guidance. He has also treated me and my pranks with utmost forbearance. Avinash Krishnan (TAMU, 2012) was an incredible support to me in searching for jobs, and I am deeply indebted to him for all his extraordinary help.

Finally and most importantly, I express most profound respect and gratitude to my mother Shafeeda Rasheed, my father Lateef Rasheed and my sister Karishma for being incredibly supportive throughout my life. I owe everything I have to them. I also thank my extended family – grandmothers, cousins, uncles and aunts – in India, for their unflinching encouragement. I thank God for all these giants on whose shoulders I have tried to stand, and for all His countless Blessings.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

# 1. INTRODUCTION

With the semiconductor industry hitting frequency and power limits on single-core chips, multi-core processors have gained wide acceptance as a means to continue performance scaling. Future many-core processors are expected to have hundreds of small processing elements working in parallel, while also interacting with each other over on-chip interconnects. Thus, the on-chip interconnect plays a crucial role in determining the performance of the chip. Shared-medium interconnects (buses) offer a simple interconnection network, ease of arbitration and straightforward support for broadcast messages. However, these interconnects are effective only for a small number of interacting components, and not suitable for the large-scale many-core chips of the future [1]. Networks-on-Chip (NoCs) offer a scalable means of communication between cores on a chip [1] [2], while also presenting several other advantages – they offer high communication bandwidth and multiple parallel communication-flows, thus increasing performance and enabling reuse of wiring resources; they lend structure to the design of a chip; and they make modular designs straightforward to implement [2]. NoC design directly impacts the performance of a many-core chip, and several studies have been made in this regard. We attempt to design an NoC router with low latency and high throughput. This section presents some fundamental concepts of NoCs, and some relevant previous work. It also introduces the contributions of this thesis, which will be explained in greater detail in later sections.

## 1.1 Background

Networks-on-Chip derive many of the characteristics of larger communication networks, but also possess certain unique characteristics. Chiefly, NoCs contain plenty of wiring resources, but lesser buffer space, compared to larger networks [2]. Several design choices impact the latency and throughput of NoCs. Some of these choices are explained briefly in the following sections.

### *1.1.1 Network Topology*

The physical layout of the network (the topology) is a fundamental design choice that determines how the cores (nodes) in the network are connected, the routing mechanism used, and the overall latency and throughput of the network. The choice of topology depends on a number of parameters, like latency, bandwidth, and power. Two-dimensional topologies are generally preferred, from a manufacturing perspective [3] [4]. The most common two-dimensional network topologies are the mesh and the torus, shown in Figure 1.1(a) and Figure 1.1(b) respectively. The mesh is a simple 2D array of nodes, with each node having a maximum of four neighbors. The torus is a mesh with its edges folded upon each other to form a donut-shaped topology. The torus reduces the average number of hops per transmission, but has slightly more complex routing and longer wire distance to travel [2] due to edge nodes being connected directly. The 2D mesh has come to be accepted as the topology of choice for NoCs, due to its simplicity and scalability, and we will use this topology for the purpose of this thesis.

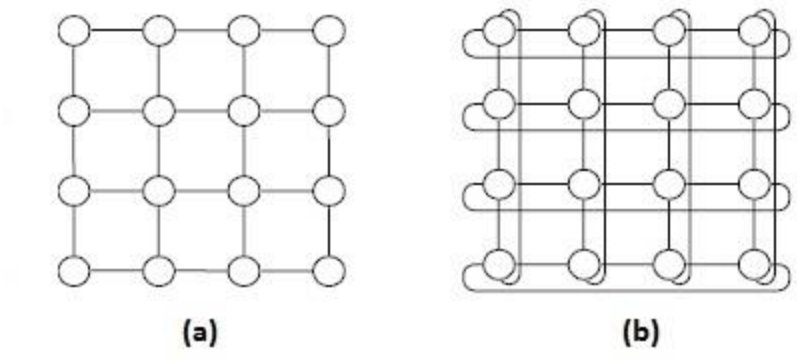


**Figure 1.1: Network topologies – 2D mesh (a) and 2D torus (b)**

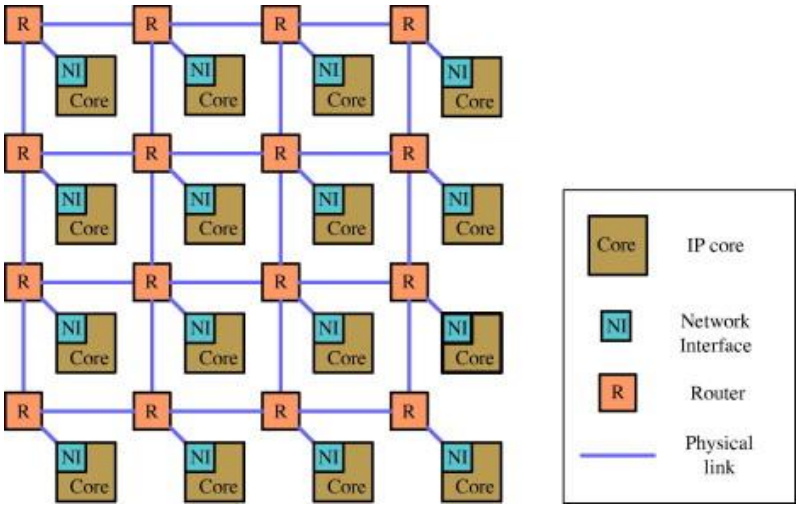


**Figure 1.2: A 4x4 mesh NoC**

Figure 1.2 shows a network-on-chip organized as a 4x4 2D mesh topology. There are 16 cores, each of which is connected to a router by a network interface. The routers are responsible for directing incoming packets to their final destinations, performing decoding and arbitration to allocate network resources to competing packets.

### *1.1.2 Packets and Flits*

NoCs are packet-switched networks. Data is transmitted through the network in the form of packets. These packets are divided into flits, which are fixed-sized units of packets and are the smallest unit of data transmission. Flit size is determined by the raw wire bandwidth of each link in the network. In this thesis, we assume that each link is 128 bits (wires) wide, and thus each flit is 128 bits wide. A packet typically consists of a header flit, body flits and a tail flit. The number of flits comprising a packet can be variable, with a minimum of one. The header flit consists of control data, such as source address, destination address, number of flits in the packet etc.

### *1.1.3 Routing*

Routing refers to the protocol used by the network to direct traffic from source to destination. NoC routing can be either *oblivious* (the routing function does not consider the state of the network) or *adaptive* (the routing decides the router based on network state information) [5]. Oblivious routing includes *deterministic* routing, with a single unchanging path between any pair of network nodes. We use *dimension-order* routing for 2D mesh networks, a type of deterministic routing; in particular, we use XY routing, a form of Manhattan routing in which packets first traverse all nodes in the X dimension, before taking a turn to the Y dimension and reaching their destination. 2D meshes employing dimension-order are arguably the most prevalent of NoC architectures due to low complexity and modularity. We exploit biases present in dimension-order routing to come up with our novel STORM router design.

### *1.1.4 Flow Control*

Flow control deals with the allocation of network resources (bandwidth and buffer space) to packets. Flow control can be bufferless, where packets are misrouted or dropped when channel bandwidth is unavailable; circuit-switched, where a control header flit first reserves all bandwidth from source to destination for a packet to travel; or buffered, where explicit flit buffers are used at the input ports/output ports of each router to store flits that wait for resource allocation [5]. Among these, buffered flow control is most efficient, as it prevents misrouting/dropping of packets and the inefficient reservation of multiple links by a single packet at once. Buffered flow control necessitates the need to have modules to allocate buffer space to flits/packets, in addition to channel bandwidth allocation modules. Among buffered flow control techniques, *virtual channel* flow control [5] [6] is the most efficient in terms of latency and network throughput. It allows unblocked packets to bypass blocked packets and utilize channel bandwidth efficiently, i.e., it removes head-of-line (HoL) blocking. Virtual channel flow control is coupled with *credit-based* flow control [5], in which each router keeps track of available downstream virtual channel buffer space by receiving *credits* (tokens) from a downstream router. When a buffer space is freed up, each router sends a credit to its upstream neighbor. The next section (Section 2) on the microarchitecture of an NoC router explains flow control and resource allocation in greater detail.

## 1.2 Contributions

Under realistic workloads, traffic patterns in typical NoC designs are known to be both biased and highly imbalanced. Figure 1.3 illustrates this effect. The figure shows average link utilization for the SPLASH-2 benchmark fft, executing in a 49-node CMP interconnected via a 7x7 2D mesh NoC under dimension-order routing. From this figure, we find that link utilization is highly imbalanced from node-to-node within the network. Further we find that there are obvious biases in the traffic flows, with straight-paths being taken more frequently than turn paths. In traditional router designs, these imbalances and biases lead to performance loss, as routers are typically designed under the assumption of uniform utilization.

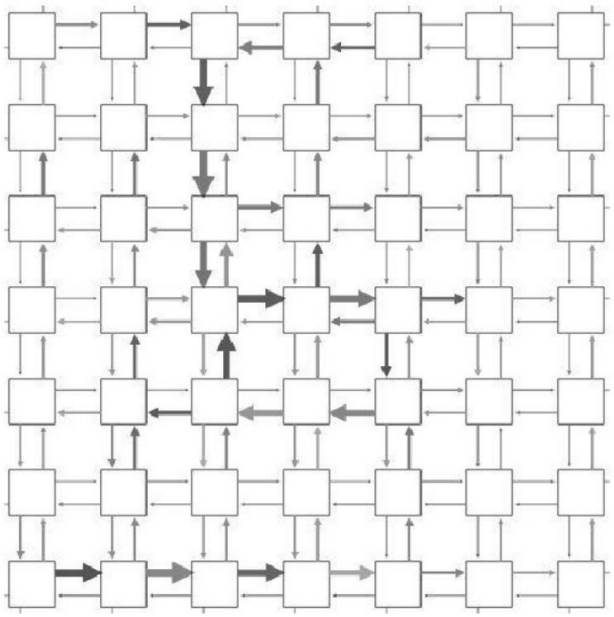

**Figure 1.3: Link utilization for the fft SPLASH-2 benchmark in a 47-node CMP interconnected via a 7x7 2D mesh NoC. Thicker and darker lines indicate higher link utilization.**

Our proposed design, *STORM* - a Simple Traffic-Optimized Router Microarchitecture - reduces the router pipeline from two stages to one, while improving clock cycle time up to 17% versus baseline, for a total reduction in zero-load latency of up to 41%. Additionally, the STORM also offers an increase of up to 14.6% in network throughput over baseline. On realistic workloads, STORM improves latency by 36-41%. Moreover, STORM consumes less area and dynamic power than a baseline router with the same number of VCs. To achieve these benefits, STORM requires a minimum of 4 VCs per input port, an acceptable number under realistic design complexity constraints.

We perform cycle-accurate simulations to compare the network-level performance of STORM with that of a baseline virtual channel router, and a router employing wavefront-based switch allocation [7] [3]. We then proceed to perform RTL synthesis of the baseline, wavefront and STORM router designs and establish that STORM can be operated at a higher frequency. We find that STORM offers a higher network throughput at lower latency for uniform-random traffic. Finally, we perform trace-based simulations of PARSEC and SPLASH2 benchmarks to evaluate STORM against the baseline and wavefront designs, and we observe consistent performance improvements while using STORM.

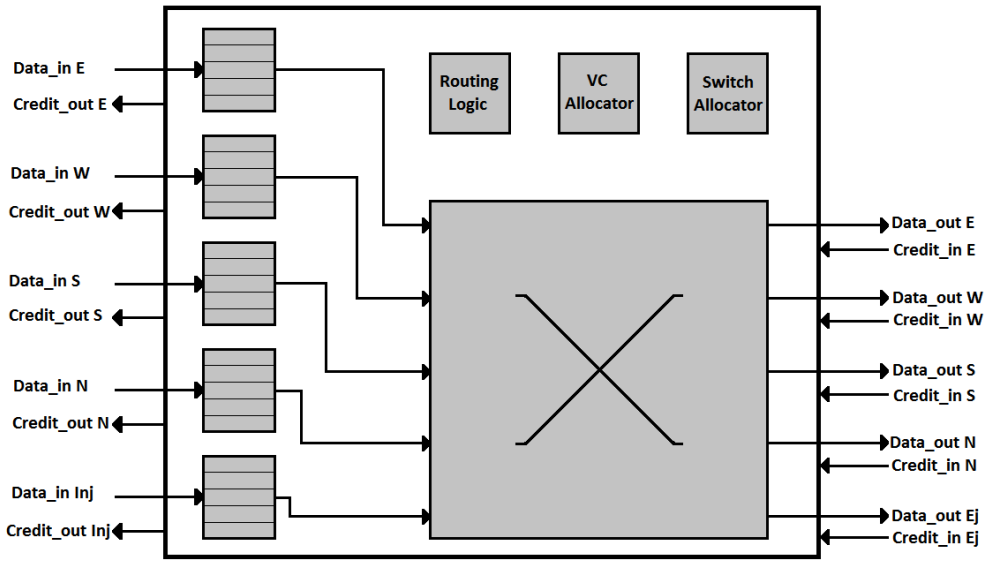


**Figure 2.1: A virtual-channel router with 5 VCs per input port**

Figure 2.1 shows the microarchitecture of a state-of-the-art *virtual channel router* employing credit-based flow control, for a 2D mesh network. There are five input ports, one for each of the cardinal directions (East, West, South and North) and one for local injection from the processing element (PE). Similarly, there are five output ports, one for each cardinal direction and one that feeds the local PE. This router is input-buffered – each input port contains a buffer structure divided into virtual channels (VCs) [6]. The use of a simple FIFO queue at each input can block flits in the queue that request available output channel bandwidth from moving forward if the output channel requested by the flit at the head of the queue is unavailable (head-of-line blocking). The

use of virtual channels allows flits requesting unblocked output ports to bypass those that request blocked outputs, thereby significantly increasing throughput.

An alternate input buffer configuration is to have separate buffers for each virtual channel, with a demultiplexer feeding incoming flits into the buffers and a multiplexer reading flits out of the buffers. However, a single buffer structure per input port is more efficient in terms of area, and not so much different in terms of timing [3].

The router contains a 5x5 crossbar switch, typically implemented using multiplexers and demultiplexers. The use of dimension-order routing (XY) removes the need for a full crossbar, as incoming flits at the North and South inputs will never request the East or West output ports. The router also contains modules for virtual channel and switch allocation, which are described in the following subsection.

## 2.1 Router Pipeline

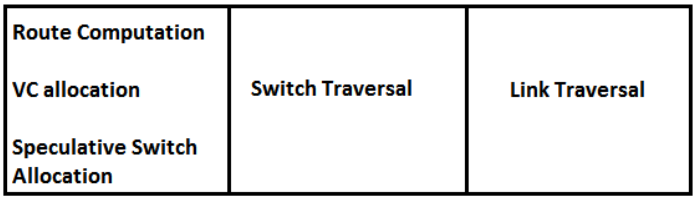


**Figure 2.2: The baseline router pipeline**

Figure 2.2 shows a typical two-stage router pipeline, with an additional stage for link traversal. The first stage involves the following functions:

### *2.1.1 Route Computation (RC)*

This function involves the computation of the output port desired for every new header flit at every input virtual channel. This information is used by the virtual channel (VC) and switch allocation modules to allocate network resources, and thus these functions cannot be performed until the output port at the current router is known. The deterministic nature of dimension-order routing enables the use of *lookahead routing* [5], where each router computes the output port at the *next* hop instead of at the current hop, and encodes this information in the header flit. Thus, at each hop, the output port required at the present node is known directly from the header, enabling VC and switch allocation to be performed, while the RC module computes the next-hop output port in parallel.

### *2.1.2 Virtual Channel (VC) Allocation*

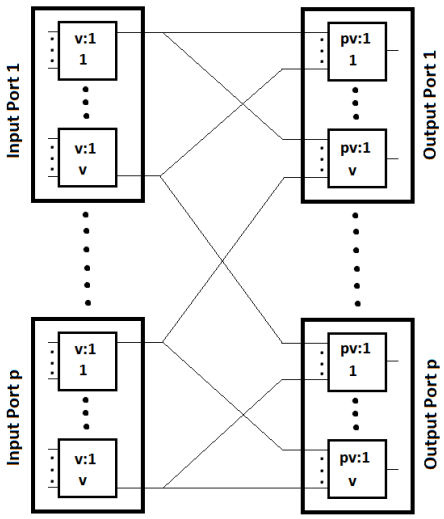


**Figure 2.3: Separable VC allocation**

Newly-arrived header flits at all input virtual channels need to be allocated downstream virtual channels. The VC allocator reads the requested output port from each header flit, and arbitrates between input VCs that request the same output VC. This allocation is done using *separable allocation* [5] [3], a two-level round-robin arbitration. Figure 2.3 shows the arbitration involved in separable VC allocation. Let us denote the number of input (output) ports by $p$ and the number of VCs per input (output) port by $v$. In the first level, every input VC selects one output VC out of $v$ VCs at its desired output port; this level involves $v$:1 arbiters for every input VC, for a total of $pv$ $v$:1 arbiters. The first level thus produces $pv$ winners. At the second level, each output VC selects one input VC out of all (potentially $pv$) input VCs that request it. Since there are $pv$ output VCs, and there can be up to $pv$ requests for each output VC (from the first level), this stage requires $pv$ $pv$:1 arbiters, as shown. Input VCs that are not allocated output VCs participate in allocation in the next cycle. Whenever a flit exits a VC to traverse the crossbar, a credit is sent to the upstream router indicating freed-up buffer space.

VC allocation (also referred to as *reallocation* due to the reuse of VCs by subsequent packets) can be *atomic* or *aggressive* [3]. In atomic VC allocation, each VC can only contain a maximum of one packet, even if there is sufficient buffer space to accommodate other packet(s) in the queue. This requires a VC to be completely empty, and all its credits returned to the upstream router, before it can be reallocated to a new incoming packet. Aggressive VC allocation involves more efficient use of buffer space by allowing VC reallocation as soon as a packet's tail flit is transmitted, thereby enabling multiple packets to be queued in the same VC. This method can be used for

deterministic, restricted-turn routing without the possibility of deadlock. Atomic VC allocation is suited for many-VC routers with plenty of (expensive) buffer space, while aggressive VC allocation is suited for routers with fewer VCs. We use aggressive VC allocation for our experiments in this thesis.

### *2.1.3 Switch Allocation*

Switch allocation allocates channel bandwidth (crossbar ports) to input VCs that have secured an output VC. This allocation, typically, is also done in two-level round-robin fashion. In the first level, one VC is chosen at each input port ($p$ $v$:1 arbiters). Since among these $p$ winning VCs, more than one can potentially contend for the same crossbar output port, a second level of arbitration ($p$ $p$:1 arbiters) is required to match a unique input VC with each output port. Once an output VC and channel bandwidth have been allocated to an input VC, the flit in that VC can traverse the crossbar in the next cycle.

Switch allocation can either be *non-speculative* or *speculative* [8]. Non-speculative allocation first waits for VC allocation to be complete, before considering for switch allocation only those VCs which have secured output VCs. When using non-speculative switch allocation, usually, an entire cycle after VC allocation is devoted to it to achieve a higher frequency of operation, at the expense of more cycles-per-hop. Speculation allows parallel VC and switch allocation in the same cycle by considering all input VCs, regardless of whether they have secured an output VC, for the switch allocation process. Candidate input VCs may or may not have output VCs allocated to

them; the winners of speculative switch allocation, if they also win VC allocation, traverse the crossbar switch in the next cycle. If a winner of switch allocation fails to obtain an output VC, it does not traverse the crossbar, resulting in an unutilized crossbar link for that cycle. In order to improve crossbar link utilization, speculative arbiters prioritize input VCs that already have an output VC over those input VCs that do not. Speculation is particularly effective under low network load, due to low contention which results in a high probability for speculative winners to obtain output virtual channels.

### *2.1.4 Switch Traversal*

This stage involves the traversal of the crossbar switch by flits from VCs which win both VC and switch allocation. These flits are latched onto registers at the crossbar output ports.

### *2.1.5 Link Traversal*

Flits latched at crossbar output ports are transmitted across the downstream link to be latched at downstream router VCs. This stage is not considered part of the router pipeline, but together with the router pipeline stages, results in a per-hop flit latency of three cycles.

# 3. RELATED WORK

The efficient design of NoCs and NoC routers has been a subject of extensive research. NoCs have come to be widely recognized as the most preferred communication infrastructures for chips with many interacting modules (Chip Multi Processors, Multi-Processor Systems-on-Chip). Dally and Towles [2] introduced on-chip interconnection networks as a replacement for global wiring for low-latency, high-bandwidth communication. Benini and De Micheli [1] proposed NoCs as on-chip "micronetworks" for scalable communication between different components of increasingly complicated Systems-on-Chip (SoCs). They present NoCs as deriving several of the desirable features of general, macro networks.

Dally [6] analyzes, using theoretical models and practical simulations, the effect of virtual channels on the throughput of general networks. Virtual channels are found to greatly reduce Head-of-Line (HoL) blocking and therefore offer significantly better utilization of physical bandwidth. Mello et al. [9] further demonstrated the usefulness of virtual channels by incorporating them into the low-latency Hermes NoC infrastructure and finding significant gains in network throughput. Virtual channels are now a standard feature of high-performance NoCs.

Bjerregaard and Mahadevan [4] survey NoCs at different levels – system, network adapter, network, and link; they also describe the salient features of existing NoCs as examples. Ogras, Hu and Marculescu [10] identify NoC research problems, such as topology, routing and switching, and present a unified view of these problems to

generate a design space for NoCs. Extending this work, Marculescu, et al., [11] provide an extensive analysis of the design challenges and performance metrics for NoCs, from the perspectives of the system, network microarchitecture and circuit complexity. They use mathematical models to characterize applications and network traffic, explore routing policies, switching protocols, network topologies and router designs, among several other contributing factors to NoC design, to come up with a comprehensive design-space exploration for NoCs. Nicopoulos [12] presents a number of innovations geared towards addressing one or more of five issues – performance, area, power, reliability, and variability – that he identifies as critical evaluation metrics for NoCs.

Among the vast work on NoCs in general, there has been particularly significant attention devoted to the efficient design of NoC routers. Peh and Dally [8] introduce speculative switch allocation for NoC routers, and a logic-effort based theoretical delay model for estimating delay in pipelined routers. Mullins, et al., [13] propose a single-cycle router implementation using simplified VC and switch allocation logic, using techniques like maintaining a queue of free VCs and the use of tree arbiters to simplify arbitration and enable a lower cycle time.

Adaptive routing is an attractive option to improve network performance when traffic patterns are non-uniform, despite the considerable overhead it imposes on network complexity. Kim, et al., [14] propose a router architecture that enables low-latency adaptive routing by grouping VCs into path sets, depending on the destination's quadrant in the network, and uses congestion information to make intelligent routing choices. To enable better load-balance across NoCs employing adaptive routing, Gratz,

Grot and Keckler [15] introduce regional congestion awareness (RCA), a low-overhead scheme to enable decisions based on congestion information spread over a greater area than just adjacent nodes. Despite offering some great advantages, we do not consider adaptive routing in this thesis and focus on simple, oblivious, dimension-order routing.

Kim, et al., [16] propose a dimensionally-decomposed router that supports both deterministic and adaptive routing. The primary feature of their work is the use of smaller, faster 2x2 crossbars with a unique switch allocation strategy that attempts to increase network throughput through maximum bipartite matching. They partition input virtual channels into path-sets based on dimensions (X or Y). As described in section 5, our STORM design partitions input virtual channels based on output ports, and uses a single level of switch arbitration and a single stage of multiplexers for the switch. Unlike STORM, Kim, et al., [16] focus on light-weight router designs with very limited VCs. Further, their design does not address circuit complexity, and thus cycle time may be sacrificed in their design relative to the baseline.

Early work by Tamir and Frazier [17], which applies to communication switches in general, proposes switches using non-FIFO, dynamically-shared input buffers, which provide higher throughput compared to FIFO/static-buffer switches. Nicopoulos, et al., [18] use a dynamic VC buffer management scheme that dispenses a variable number of VCs per port on demand to increase buffer utilization and network throughput.

Kumar, et al., [19] propose a high-frequency single-cycle router that uses a low-latency, non-speculative VC/switch allocation mechanism. The single-cycle operation is

achieved using advanced control bundles that race ahead of data packets and remove control setup from the router's critical path.

Kim [20] proposes a low-cost router microarchitecture that uses a dimensionally-partitioned crossbar with intermediate buffers, for dimension-order routing. Switch arbitration is simplified by prioritizing "in-flight" packets over packets waiting to be injected. If packets wait for too long to be injected, the router sends an explicit signal to its upstream router to stop its flow of "in-flight" packets, thereby preventing starvation. Ramanujam, et al., [21] introduce a router design that emulates an output-buffered router (OBR) using a distributed shared-buffer (DSB) scheme to obtain higher throughput compared to a traditional input-buffered router (IBR), but at the expense of increased pipeline depth (five stages compared to three for IBR).

Becker [3] provides a comprehensive examination of the nature, advantages and disadvantages of various arbiter and allocator implementations, VC allocation, switch allocation and buffer management. He describes combined VC and (speculative) switch allocation to reduce allocator design complexity. A similar, non-speculative, combined VC and switch allocator is described for low-speed FPGA NoCs by Lu, McCanny and Sezer [22]. Zhao, Zhang and Yang [23] introduce a novel, low-latency switch-arbitration mechanism for many-VC routers, which exploits high VC occupancy by implementing shorter arbitrations compared to a traditional arbiter.

### 3.1 Wavefront Switch Allocation

In addition to the baseline NoC router, we compare our STORM design with a router employing *wavefront* switch allocation [7] [3]. Wavefront allocation operates on the *request matrix*, a Boolean matrix indicating requests from input ports for output ports. If $P$ is the number of ports (input or output), the request matrix $R$ is a $P$ x $P$ matrix with $R_{ij} = 1$ if there is a request from at least one VC at input port $i$ for output port $j$, and $R_{ij} = 0$ otherwise. The result of the allocation is a *grant matrix* $G$, another $P$ x $P$ matrix with $G_{ij} = 1$ if input port $i$ is granted access to output port $j$, and $G_{ij} = 0$ otherwise.

In each cycle, wavefront allocation starts with one highest-priority diagonal on the request matrix, grants all requests on this diagonal, and nullifies other requests on the same row and column as each granted request. It then proceeds to another diagonal and repeats the process until all $P$ diagonals are covered. Since all elements on any given diagonal represent non-conflicting requests, they can be granted at once before proceeding to the next diagonal.

An illustration of the operation of a wavefront allocator can be seen in Figure 2.4. Each panel represents one cycle of operation. The highest-priority diagonal in each cycle consists of the grey boxes. At each cycle, a granted request is shown in green, and a declined request is shown in red. Here, the priorities of the diagonal are updated in circular fashion, with the diagonal immediately after the current highest-priority one getting top priority in the next cycle. We assume that the request matrix remains the same for the four cycles shown. This simple scheme can cause unfairness when the request matrix is sparse, leading to some requests being granted more often than others.

This can be seen from Figure 2.4, as the request from the West input to the South output is granted 4 times out of 5 cycles.

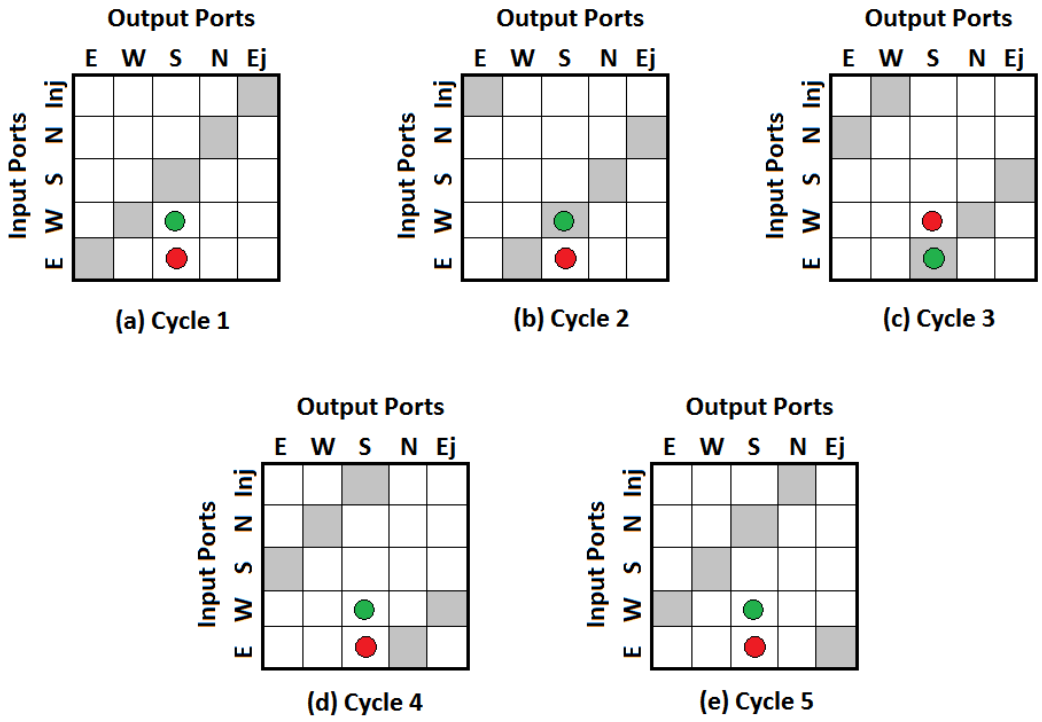


**Figure 2.4: Wavefront allocator operation – request matrices in each cycle (green: request granted; red: request declined)**

Becker [3] addresses this fairness issue by introducing a modified priority update mechanism, in which the highest priority diagonal for the next cycle is the one following the current top priority one which had any requests at all (and not necessarily the immediate successor of the current top priority diagonal). We call this modified wavefront scheme *wavefront*$^{+}$, henceforth.

# 4. AN ANALYSIS OF CONTENTION

Contention occurs in NoCs when multiple entities (flits) compete for the same resource (buffer space or bandwidth). Contention necessitates arbitration between flits to allocate buffer space or channel bandwidth. Desirable features of arbitration are high speed, fairness, and ensuring the best utilization of available resources. The last feature of ensuring high resource utilization directly results in higher network performance (throughput). In this section, we analyze and classify contention for channel bandwidth, and present network-level simulation results to analyze the degrading effect of contention on network throughput. This analysis will help put in context our motivation behind the STORM router microarchitecture that we propose in section 5.

## 4.1 Types of Contention

Let us consider contention between flits for channel bandwidth in a typical NoC router. We identify two types of contention – *degrading* and *non-degrading*. *Degrading* contention is that which results in channel bandwidth being unutilized at any point of time. *Non-degrading* contention is that which, even when present, does not impact channel bandwidth utilization.

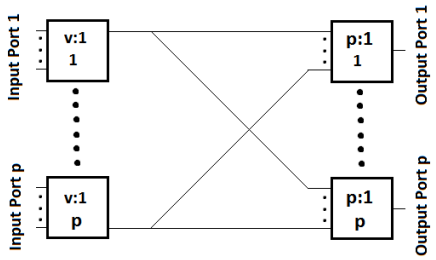


**Figure 4.1: Two-level round-robin switch allocation**

Section 2.1.3 introduced switch allocation in a typical NoC router as a two-level, separable round-robin allocation. Figure 4.1 illustrates this mode of allocation. In the first level, one VC from each input port is chosen to contest for its desired output port. In the second level, for each output port, one of the (possibly) many first-level winners that contest for that output port is chosen.

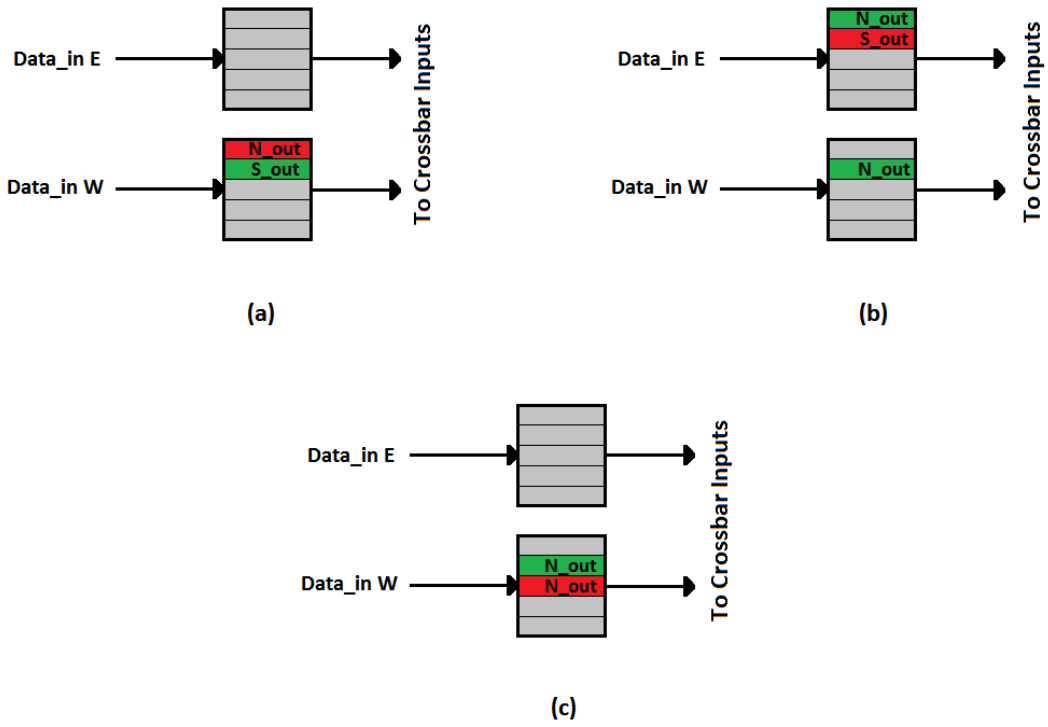


**Figure 4.2: Illustration of contention – First-level contention (a), Second-level contention (b), and non-degrading contention (c)**

Figure 4.2 shows how the presence of (degrading) contention can result in channel under-utilization. We identify two levels of such contention. First-level contention occurs when different VCs at the same input port request different output ports (Figure 4.2a), but only one of these requests can be granted, and the output port

requested by the losing VC remains unutilized. Second-level contention occurs when winners of the first level of arbitration contend for the same output port, and thus only one of these first-level winners can win the second-level. This is illustrated in Figure 4.2b by the two VCs at different inputs which both request the North output, and both win first-level arbitration; if the first-level winner from the Eat input was the other VC (which requests the South output), both North and South output ports could have been utilized in the same cycle. These two levels of degrading contention result in under-utilization of the crossbar switch. Essentially, this loss in crossbar throughput can be viewed as a consequence of inefficient matching between input ports and output ports.

Contention can also be non-degrading, as in Figure 4.2c, where two VCs from the same input port contend for the same output port; picking either VC in the first level of arbitration will result in the same crossbar utilization.

The efficient matching of input ports with output ports is a *maximum bipartite matching* or *maximum cardinality matching* problem. It requires us to grant as many switch requests from input VCs as possible (with the number of output ports setting the upper bound on the possible number of grants), subject to the constraint of choosing only one VC from each input port. While theoretical algorithms exist to solve this problem (like the augmenting-path algorithm), their realization in hardware involves high design complexity and is unsuitable for low-latency operation. Separable, two-level round-robin allocation provides reasonable matching efficiency at low latency, and is thus a good choice for switch allocation.

Maximum bipartite matching prioritizes crossbar utilization over fairness to input requests [3]. As a result, some flits have to wait a long time, even be starved, if there are other requests that can result in maximum use of the crossbar. Despite high design complexity and unfairness, the use of maximum bipartite matching provides us with a theoretical bound on cycle-level network performance.

Maximum bipartite matching is still constrained by the fact that only one VC can be selected from each input port (first-level contention). Maximum network throughput is obtained when all input VCs participate in arbitration for all output ports. In the case of a *p*-input router with *v* VCs per input port, this ideal case transforms the crossbar from a *p* x *p* switch to a *p***v* x *p* switch (which can be realized as *p* multiplexers with *p***v* inputs each). Arbitration in this case is single stage, with *p* round-robin arbiters (*p***v*:1) per output port. This *unrestricted* model [13] resolves both first and second level contention, and sets the upper bound on network throughput for a given amount of buffer space. However, a realistic implementation of an "unrestricted" router will have very high cycle-time, due to the large size of the switch and arbiters.

The following section provides data from cycle-accurate simulations to demonstrate throughput loss due to (degrading) contention.

### 4.2 Simulations

Cycle-accurate network-level simulations are carried out using Ocin_Tsim [24]. We simulate router performance for an 8x8 2D mesh network employing dimension-order (XY) routing on uniform random traffic. We assume a two-stage router pipeline

with an additional link traversal stage, as in section 2.1. Simulations are run for various network loads (injection rates), for 1 million 4-flit packets with a 10000 cycle warm-up phase.

Five 4-flit VCs are employed per input port. Separable, aggressive VC allocation (section 2.1.2) is used. For the baseline router, we use separable round-robin switch allocation; the maximum-matching simulations use maximum bipartite matching, while the "unrestricted" router uses single-stage allocation with bigger allocators and a bigger switch, as described in section 4.1. The wavefront$^{+}$ router uses the wavefront$^{+}$ switch allocation scheme described in section 2.2.

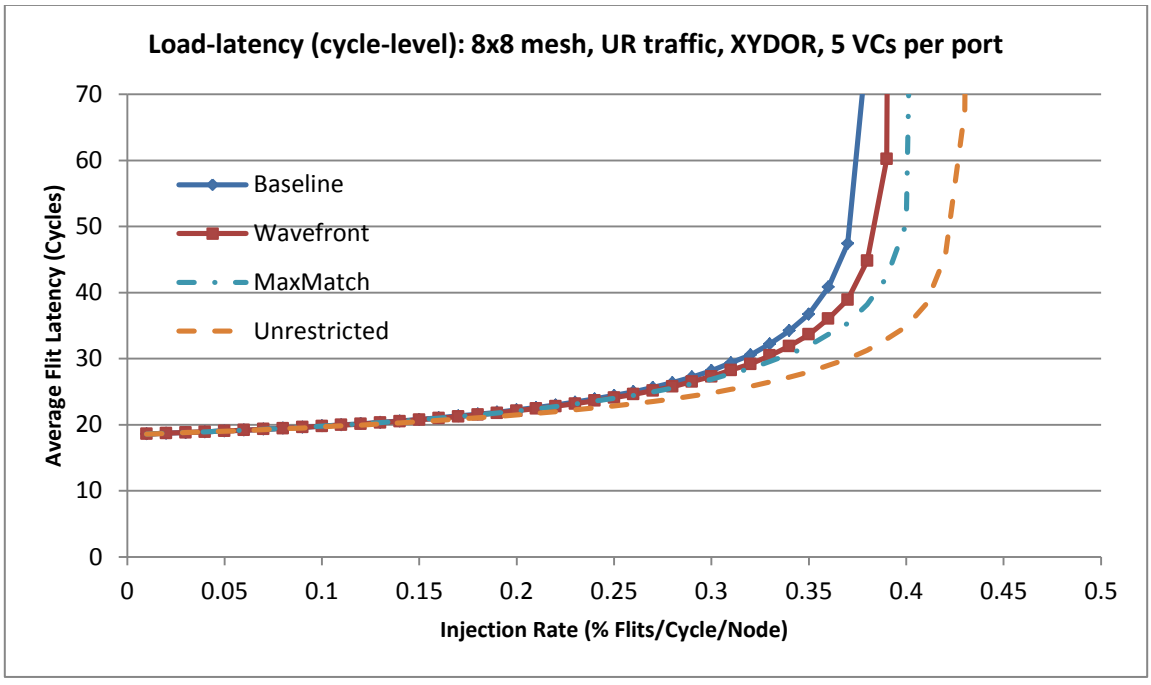


**Figure 4.3: Cycle-level load-latency curves for the baseline router, a router with maximum matching, and an unrestricted router**

Figure 4.3 shows cycle-accurate, network-level load-latency curves for the baseline router, the wavefront$^{+}$ router, the maximum-matching router and the unrestricted router. The saturation throughput (0.373 flits/cycle/node injection) of the baseline router is affected by both first and second level contention. While the wavefront$^{+}$ router still suffers from degradation caused by both levels of contention, it benefits from a wider view of the request matrix to provide a higher saturation throughput of 0.387 flits/cycle/node. Maximum matching removes the effect of first level contention and improves the saturation throughput to 0.4 flits/cycle/node injection. The unrestricted router nullifies the degrading effect of both first and second level contention, and increases the saturation throughput of the network to 0.42 flits/cycle/node.

Our proposed STORM router model, described in section 5, attempts to neutralize the degrading effect of contention by taking advantage of biases inherent to dimension-order routing and uniform random traffic. As a result, network throughput is enhanced in relation to the baseline router. More significant benefits of our design are a reduced cycle-level latency due to a shorter pipeline, and reduced timing-level latency (higher frequency) due to ease of arbitration.

## 5.1 Microarchitecture

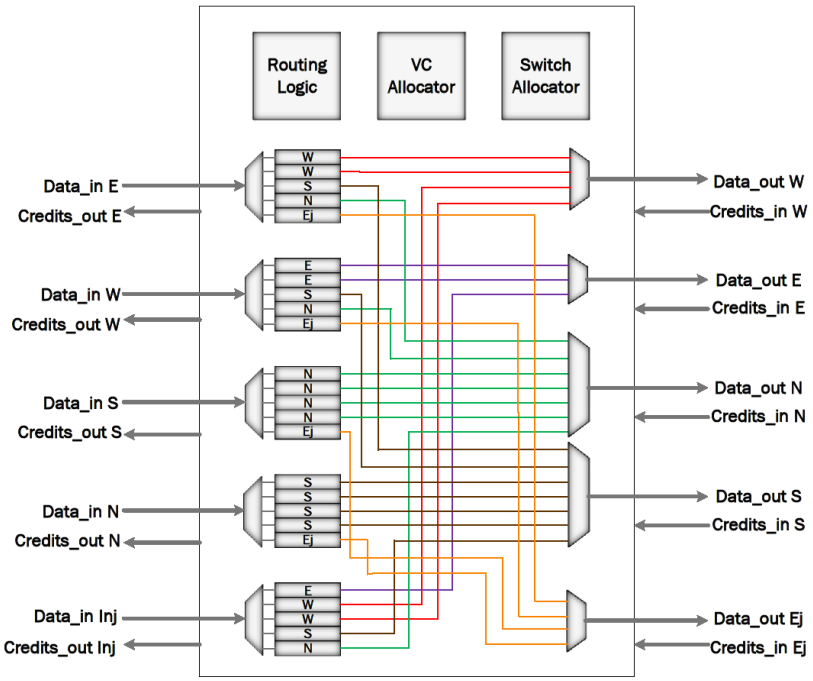


**Figure 5.1: STORM microarchitecture (sample)**

Figure 5.1 shows a sample microarchitecture of a STORM router. The standout feature of this design is the classification of input VCs based on output ports. A VC assigned to a particular output port cannot contain flits destined for other output ports. The set of VCs across all input ports that are assigned to the same output port form a *path-set*. Since there are five output ports in a router for a 2D mesh, there are five path-sets, one for each output port (East, West, South, North and local ejection).

The concept of VC partitioning to form path-sets has been explored previously - for instance, partitioning has been based on destination quadrants [14] and X-Y

dimensions [16]. As discussed in Section 3, these works, however, primarily focus on low-overhead, low-throughput networks and employ allocation techniques which require greater complexity than the baseline router, impacting cycle time. Further, they do not provide a mathematical framework to perform VC partitioning, and use the same router microarchitecture throughout the network. STORM leverages traffic pattern biases to partition VCs based on output ports; VC partitions in STORM are non-uniform across network locations and customized for location-specific traffic patterns.

Knowing that flits at each input port can request only a limited number of output ports enables us to partition the buffer space at each input port into destination-specific VCs. This partitioning is done based on the statistical nature of uniform-random traffic. Assuming a fixed number of VCs per input port, input buffers are partitioned into destination-specific VCs based on probabilistic proportions of output port requests at any given input port.

Let $V$ be the number of VCs per input port. For simplicity, we assume that $V$ is the same for all input ports in all routers across the network. We now focus on the router at any particular node. For any given router, we identify the following parameters:

- $i$ – input port index
- $P_i$ – set of output ports that can potentially be requested by $i$
- $p$ – output port index, $p \in P_i$
- $N_p$ – number of network nodes that can be reached via output port $p$
- $d_{ip}$ – number of VCs at input port $i$ which request output port $p$

Our goal is to find $d_{ip}$ for all $i$ and $p$, for the network node under consideration. We assume that all possible final destination nodes for any flit at any input port are equally likely – a characteristic of uniformly random traffic patterns. Thus, the splitting of the $V$ input VCs at each input port is straightforward, and is given by

$$d_{ip} = \left\lceil \left( \frac{N_p}{\sum_{p \in P_i} N_p} \right) * V \right\rceil \quad , \quad \forall\, p \in P_i, \quad \forall\, i$$

subject to

$$d_{ip} \geq 1 \quad , \quad \forall\, p \in P_i, \quad \forall\, i$$

and

$$\sum_{p \in P_i} d_{ip} = V \quad , \quad \forall\, i$$

We look at the specific case of the node in red in the 8x8 mesh of Figure 5.2. It is a central node and thus experiences the most traffic in a uniformly distributed random traffic pattern. Flits arriving at the East input port of the router at this node have 32 possible final destinations – 24 to the West, 4 to the North, 3 to the south, and 1 destination being the red node itself. The input VCs at the East input of the STORM router at this node are thus partitioned in this ratio – 24:4:3:1 – with at least one VC for each output port. For 5 input VCs per port ($V$ = 5), at the East input port we have 2 VCs for the West output, 1 for South, 1 for North and 1 for local ejection. A similar breakdown is obtained for the other input ports of this router.

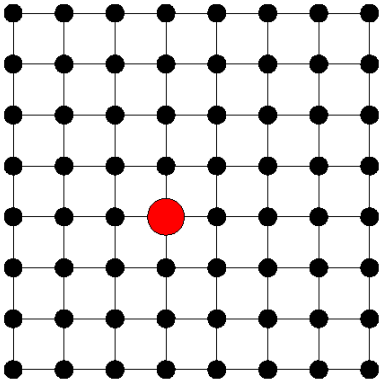

**Figure 5.2: An 8x8 mesh with a central node shown in red**

This partitioning is performed for all routers at the other nodes in the network. Thus, the microarchitectures of all routers in the network, while similar in principle, are not identical. Statistical buffer partitioning depending on node location enables us to achieve higher network throughput.

We note that this approach does lead to increased physical design and validation costs, however, routers for the entire network can be designed by left-right and top-bottom mirroring of a limited number of router layouts. Specifically, in an 8x8 network, symmetricity enables us to design the entire network using the layouts of routers in any one quadrant. Among the 16 STORM routers in one quadrant, there are 14 microarchitectures that are distinct from each other. Thus, it is possible to design and validate a 64-node, 8x8 STORM network by designing only 14 routers that are very similar to each other.

Besides this non-uniform STORM design, in Section VI, we evaluate a STORM variant that uses a single microarchitecture - customized for a high-traffic central node -

throughout the network, thus saving design and validation time. This uniform STORM design still offers lower latency and higher throughput than a baseline router design.

The classification of input VCs into destination-based path-sets enables us to introduce novel, performance-improving features in our router, as described in the following subsections.

## 5.2 Ease of Arbitration

The typical baseline NoC router employs two-level separable arbitration for both VC and switch allocation. In STORM, since we clearly define path-sets, arbitration is even simpler than separable allocation and is free from degrading contention (section 4.1). The simplicity of arbitration in STORM is its most important advantage, as it directly improves the frequency of operation.

### *5.2.1 VC Allocation*

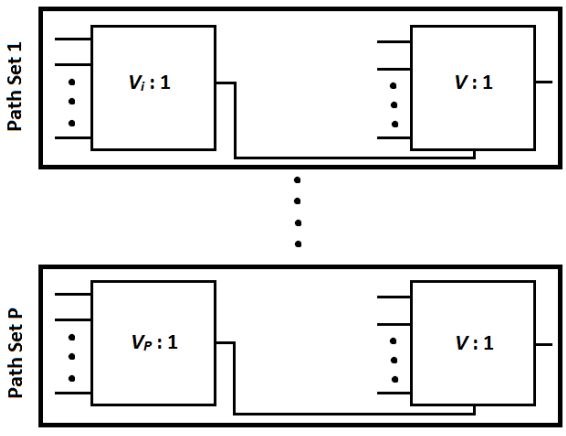


**Figure 5.3: VC allocation in STORM**

Figure 5.3 shows the VC allocation mechanism in STORM. This is contrasted with the baseline VC allocation scheme from Figure 2.3. The proposed scheme uses five sets of independent two-level arbiters, one for each path-set. If $V_i$ is the number of VCs in path-set *i*, the first-level arbiters in VC allocation perform a $V_i$:1 reduction to choose one winner from each path-set that will request an output VC (at the output port associated with that path-set). If there are *V* downstream VCs per output port, the second level of arbitration performs a *V*:1 reduction to allocate one of *V* possible downstream VCs to the winning VC from the first level, depending on its next-hop destination.

Compared to the baseline VC allocation scheme from Figure 2.3, in which the largest arbiter size is *P***V*:1 (*P* being the number of ports), the largest arbiter in the proposed scheme is one of the five $V_i$:1 arbiters.

### *5.2.2 Switch Allocation*

Switch allocation in the STORM involves simple, single-level arbitration. For each path-set *i*, there is a single $V_i$:1 round-robin arbiter (where $V_i$ is the number of VCs in path-set *i*) to pick one winner from each path-set. Contention for any given output port can occur only within its corresponding path-set, and the winner of arbitration is guaranteed the use of that output port (channel) – provided it has also acquired a downstream virtual channel. Thus, contention in STORM is *non-degrading*, i.e. it does not affect network throughput. This switch arbitration scheme, while simple in terms of logic complexity, is also fair.

### 5.3 Reduced Pipeline Depth

The proposed STORM design (Figure 5.1) employs a single level of multiplexers that serve to connect input VCs to output ports, instead of the more complicated crossbar switch that can be seen in a typical router. A typical router pipeline consists of two stages (Figure 2.2), the second of which is *switch-traversal*. The simple nature of the "switch" in STORM, together with simplified arbitration, enables us to combine this switch traversal stage with the first pipeline stage (arbitration) without affecting the critical path too much, thus enabling single-cycle router operation, with an additional link traversal cycle. This is a significant advantage of STORM, as it reduces the zero-load latency of the network by 33% in terms of raw cycles. In section 6, we evaluate this single-cycle design and the more traditional two-cycle design for STORM.

### 5.4 Other Features

The use of path-sets introduces the necessity for each router to know not only the output ports for input VCs at its own node, but also at the requested downstream node, so that it can allocate the appropriate downstream destination-tagged VC to a requesting packet. Thus, *two-hop lookahead routing* is employed, in which each router computes the output port at the next hop and also at the hop after, encoding these two output ports into the head flit. Synthesis results show that this slightly more complex lookahead routing does not lie on the critical path of the design, and thus does not affect the maximum possible operating frequency.

A feature of STORM that appears less attractive in contrast with the typical router is the reduced flexibility in VC allocation. In a baseline router, any input VC can be assigned to any incoming flit, while in STORM, only certain input VCs can be assigned to certain flits. This restricted VC allocation appears detrimental to overall network throughput at first glance, but the absence of degrading contention due to path-set based arbitration helps us better the throughput of the baseline router, as can be seen in the simulation results of Section 6.

Since we can potentially read from multiple VCs at the same input port simultaneously, we implement each VC as a buffer on its own, with a read port and a write port. Consolidation of all input VCs at one port into a large buffer structure (as is possible in the typical case) will necessitate multiple read ports from the larger buffer, which may affect the frequency of operation. Thus, we opt for individual buffers for each VC with input demultiplexing and output multiplexing.

Section 6 provides experimental data that shows the benefits of the proposed STORM router design.

# 6. EXPERIMENTAL RESULTS

We evaluate STORM at two levels: the *cycle-accurate* level, which involves the use of a cycle-accurate simulator to analyze latency and throughput in terms of cycles without regard to circuit complexity, and at the *timing* level, which involves obtaining the minimum cycle time at the nanosecond scale through synthesis of Verilog RTL implementations. For the cycle-accurate simulations we use the network simulator Ocin_Tsim [24], a parameterized C++ implementation to simulate network performance. For the RTL synthesis, we use an open-source, parameterized Verilog RTL package of the typical router [3], which also includes the description of a wavefront$^{+}$ switch allocator. We modify this RTL to describe and synthesize the proposed router design. Finally, we execute PARSEC benchmark traces using the Netrace [25] library integrated with Ocin_Tsim, along with static 7x7 SPLASH2 traces, to evaluate the STORM router against the baseline and wavefront$^{+}$ designs.

## 6.1 Cycle-accurate Simulations

Network-level simulations are performed to evaluate the typical (baseline) router, the wavefront$^{+}$ router and the STORM designs. We evaluate three variants of STORM.

- STORM-2: a 2-stage pipeline with a dedicated switch traversal stage.
- STORM-1: a single-stage router pipeline.
- STORM-1S: STORM-1 utilizing a single STORM router microarchitecture across the entire network.

Also, to serve as performance bounds, a router with maximum bipartite matching and an unrestricted router (section 4.1) are also simulated at the cycle-accurate level.

Simulations are carried out on an 8x8 2D mesh network with dimension-order (XY) routing using a uniform-random traffic pattern, for varying network loads (injection rates). We simulate the designs with 5, 6, and 7 VCs per port, with each VC being 4 flits deep. The runs last until 1 million 4-flit packets have been injected into the network and been routed to their destinations, with a warm-up phase of 10000 cycles. Flit latencies are recorded and plotted against offered loads.

Figures 6.1 to 6.3 show the cycle-level load-latency curves for the different designs under consideration, with the number of VCs per port ranging from 5 to 7. It can be seen that at the cycle-level, both STORM-2 and STORM-1 outperform the baseline router and the wavefront$^{+}$ routers in terms of throughput; in the case of 7 VCs, the STORM designs even approach the throughput of the high-latency, unrealistic "unrestricted" design, which sets an upper bound on the saturation throughput of the network (for a two-stage design).

We also perform cycle-level simulations by increasing the number of VCs per port from 4 to 16 (Figure 6.4). It can be seen that STORM-1 and even STORM-1S consistently outperform the baseline and wavefront$^{+}$ designs in terms of throughput; in fact, STORM-1 approaches the unrestricted case when the number of VCs is sufficiently high. With a sufficiently high number of VCs, STORM-1 manages to outperform the unrestricted router, as reduced pipeline depth offers lower latency and thus better throughput.

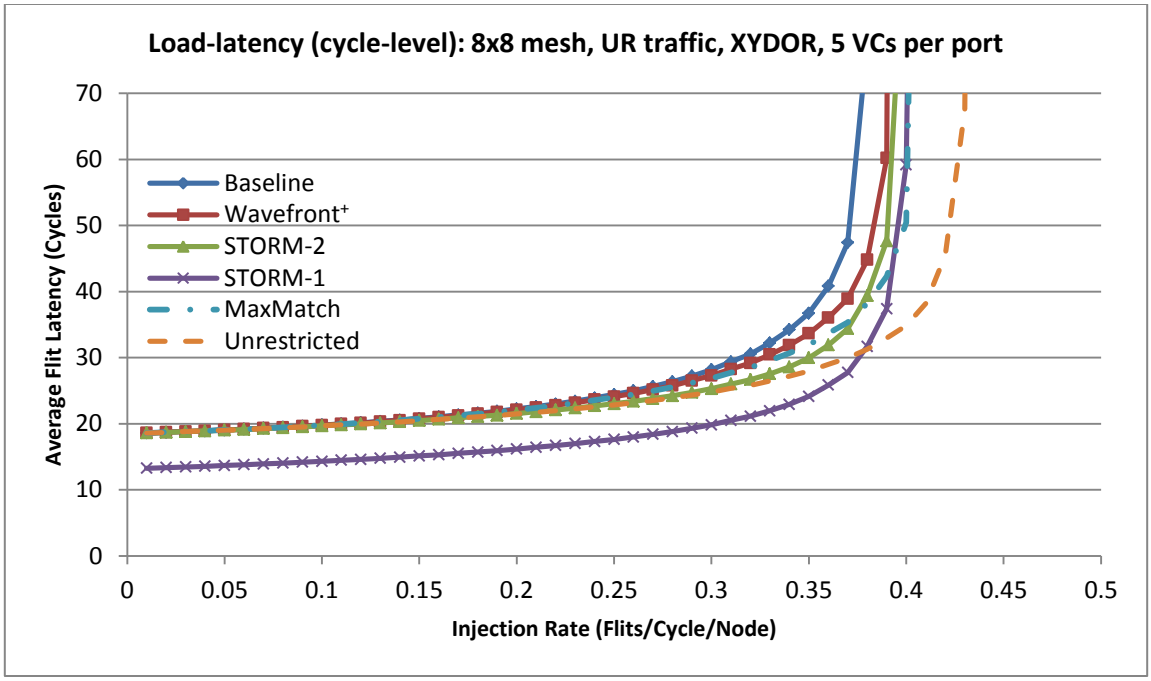


**Figure 6.1: Load-latency curves (cycle-level) – Uniform Random – 5 VCs per port**

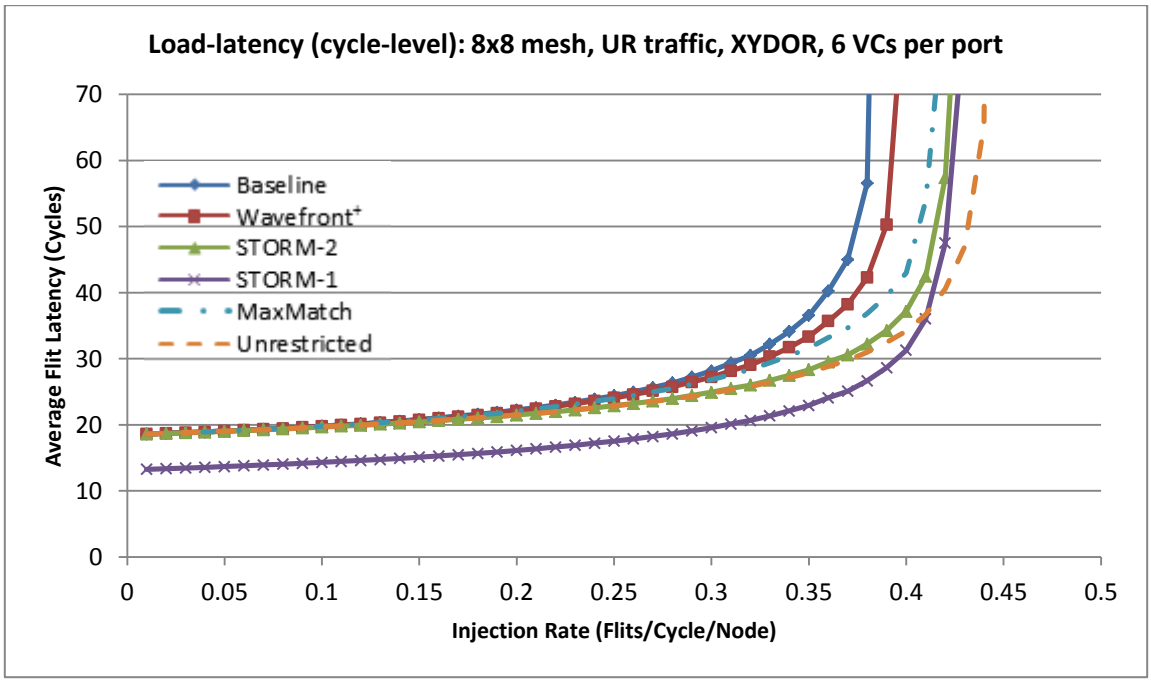


**Figure 6.2: Load-latency curves (cycle-level) – Uniform Random – 6 VCs per port**

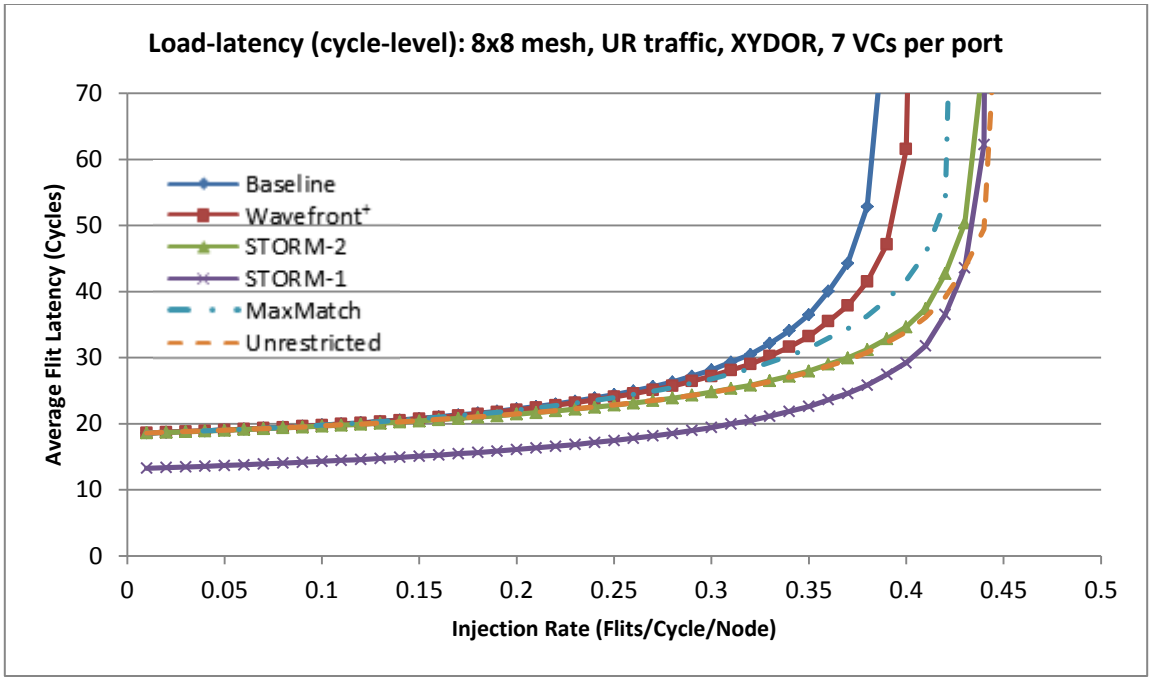


**Figure 6.3: Load-latency curves (cycle-level) – Uniform Random – 7 VCs per port**

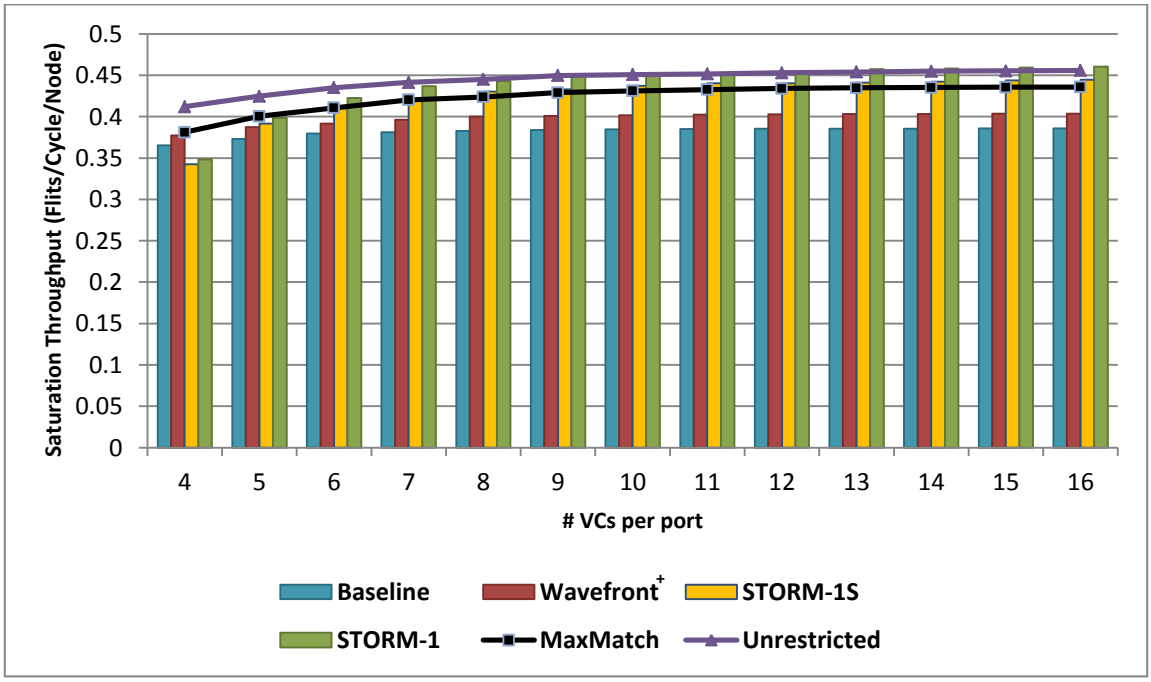


**Figure 6.4: Scaling of saturation throughput for uniform random traffic with an increasing number of VCs (throughput measured at latency = 56 cycles)**

The STORM design requires a minimum of 4 VCs per input port to function – one for each possible output port. This 4-VC configuration performs worse than the baseline and wavefront$^{+}$ designs, due to head-of-line blocking. However, as the number of VCs increases, STORM offers significant throughput benefits.

We also attempted to simulate, at the cycle-accurate level, a version of the RoCo router proposed by Kim, et al., [16]. This router partitions input virtual channels based on dimensions (X or Y) – there are input VCs that can only hold flits that request the East/West outputs, and those that can only hold flits that request the North/South outputs. There are two small 2x2 crossbars, one for the X dimension and one for Y, with a "mirror" allocator for switch allocation. In addition, this router employs "early ejection" to eject flits at their destination before resource allocation and switch traversal, saving two cycles at the destination. We simulated a baseline router, the RoCo router and our STORM-1S design using five 4-flit deep VCs per input port. The RoCo was found to perform marginally worse than our baseline router in terms of throughput (3.1% lower); STORM-1S exceeded the throughput of our RoCo implementation by 10.5%. Although the RoCo is expected to offer higher throughput than the baseline router, artifacts in our implementation, such as inefficient input VC partitioning, unfair priority updates and unfair switch allocation could be the reasons behind the poor performance of our RoCo implementation. We do not proceed to implement RoCo on RTL; however, we expect it to have a higher cycle time than STORM due to its more complex switch allocation process.

### 6.2 RTL Synthesis

For STORM, the router microarchitecture varies across the network, depending on node location. This leads to variable VC organization, and by extension, variable arbiter and switch sizes, across the network. To the first order, we estimate the most complex (slowest) STORM router to be the one with the largest path-set size across the network. The path-set size directly impacts arbiter and multiplexer sizes, and thus the router(s) with the largest path-set size can be assumed to set the maximum operating frequency of the network.

Figure 6.5 shows a map of the maximum path-set size at each node in an 8x8 mesh network implementing the STORM design, for 5, 6 and 7 VCs per port. It can be seen that in all three cases, the worst-case nodes with the largest path-sets are at the edges of the network. For each case (5/6/7 VCs per port), we implemented on RTL a router containing a path-set that is as large as the largest in the network. It should be noted, however, that since there are only 4 edge nodes with the biggest path-sets, their VCs can be reorganized (maximum path-set size reduced) to achieve higher clock frequency without significantly impacting throughput.

We implemented the STORM design (STORM-2 and STORM-1) by adapting the open-source RTL package by Daniel Becker from Stanford University [3]. This parameterized RTL package includes implementations of a traditional, baseline virtual channel router, and different types of allocators, including the wavefront$^{+}$ allocator. We synthesize the baseline, wavefront$^{+}$ and STORM designs using Synopsys Design Compiler (Ultra), increasing the synthesis clock frequency constraint from 1 GHz in

steps of 50 MHz until a timing slack violation is reported. The library used is TSMC's TCBN45GSBWP 45nm CMOS library with an operating voltage of 0.9 V, and an FO4 delay of 34.6 picoseconds [3].

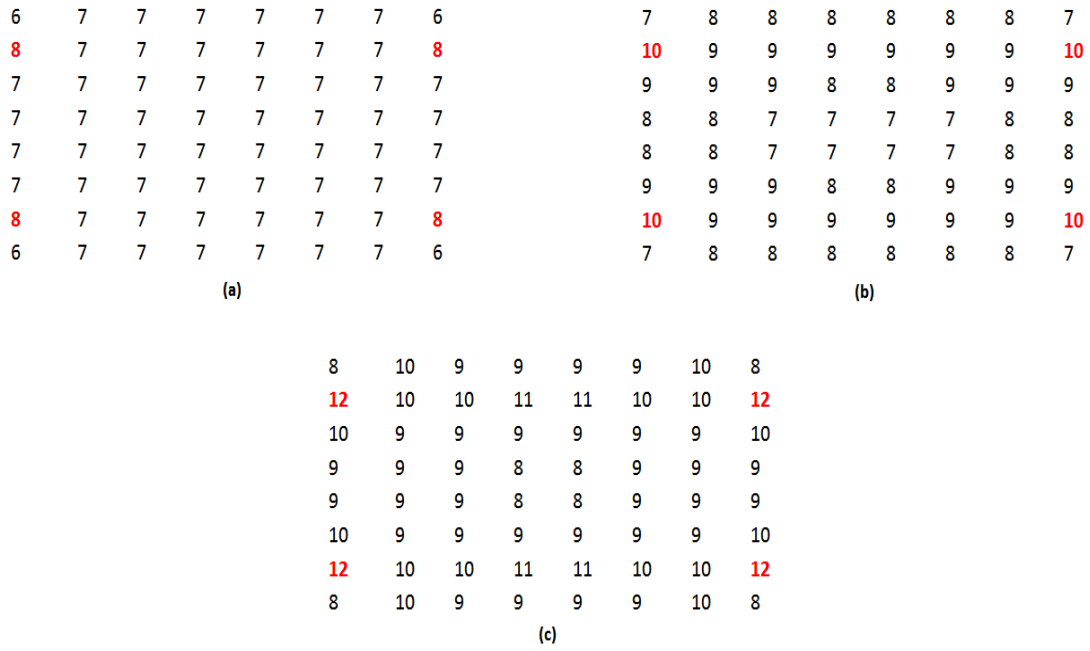


**Figure 6.5: Maps of maximum path-set size for STORM at each node in an 8x8 mesh network – for 5 VCs per port (a), 6 VCs per port (b), and 7 VCs per port (c).**

The minimum cycle time, and the corresponding maximum frequency, for each design are shown in Table 6.1, for 5, 6 and 7 VCs per port. Table 6.2 shows the area consumed by each design and the total dynamic power (at Design Compiler's default 50% activity factor) when synthesized at a clock frequency of 1 GHz.

| | Design | Max. Frequency (GHz) | Min. Period (ns) | Min. Period (FO4) |
|---|---|---|---|---|
| **5 VCs** | Baseline | 1.55 | 0.65 | 18.65 |
| | Wavefront$^{+}$ | 1.25 | 0.80 | 23.12 |
| | STORM-2 | 2.05 | 0.49 | 14.10 |
| | STORM-1 | 1.80 | 0.56 | 16.06 |
| **6 VCs** | Baseline | 1.45 | 0.69 | 19.93 |
| | Wavefront$^{+}$ | 1.25 | 0.80 | 23.12 |
| | STORM-2 | 2.00 | 0.50 | 14.45 |
| | STORM-1 | 1.75 | 0.57 | 16.52 |
| **7 VCs** | Baseline | 1.45 | 0.69 | 19.93 |
| | Wavefront$^{+}$ | 1.25 | 0.80 | 23.12 |
| | STORM-2 | 1.70 | 0.59 | 17.00 |
| | STORM-1 | 1.40 | 0.71 | 20.64 |

**Table 6.1: Minimum clock period for the baseline, wavefront$^{+}$ and STORM router designs, for 5, 6 and 7 VCs per input port**

| **Design** | **Area (mm$^2$)** | | | **Dynamic Power (mW)** | | |
|---|---|---|---|---|---|---|
| | *5VCs/port* | *6VCs/port* | *7VCs/port* | *5VCs/port* | *6VCs/port* | *7VCs/port* |
| Baseline | 0.125 | 0.150 | 0.176 | 60.464 | 71.628 | 82.526 |
| Wavefront$^{+}$ | 0.126 | 0.153 | 0.179 | 60.866 | 72.796 | 83.631 |
| STORM-2 | 0.113 | 0.143 | 0.166 | 56.934 | 71.307 | 82.281 |
| STORM-1 | 0.112 | 0.143 | 0.166 | 56.706 | 71.13 | 82.316 |

**Table 6.2: Area and dynamic power for the baseline, wavefront$^{+}$ and STORM router designs, operating at 1 GHz, for 5, 6 and 7 VCs per input port**

The minimum cycle time (in FO4 delays), and the area and power consumed at 1 GHz are shown graphically in Figures 6.6 to 6.8.

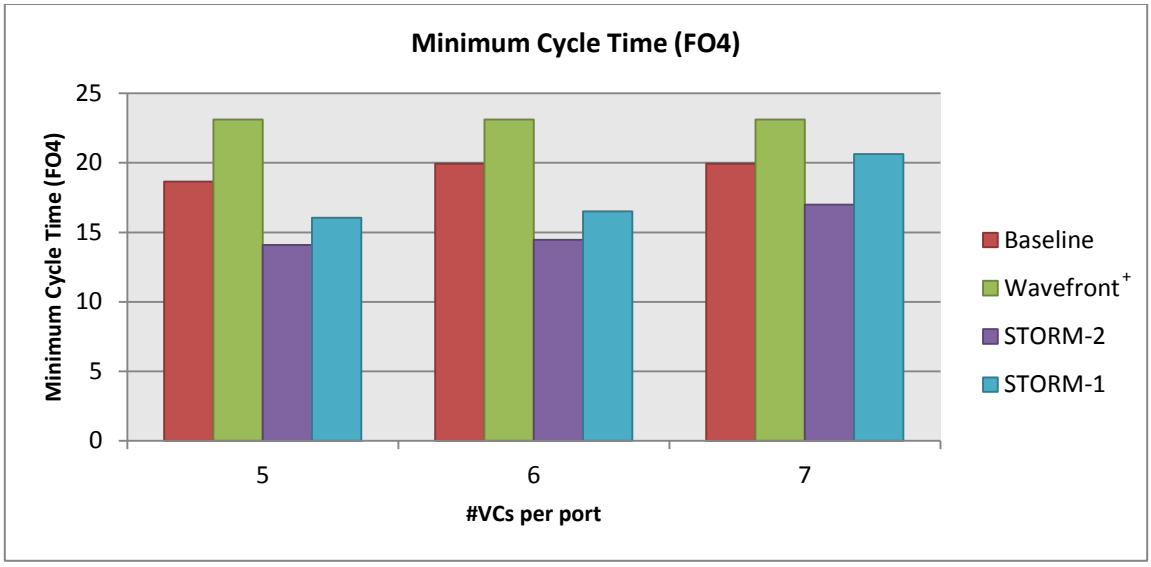


**Figure 6.6: Minimum cycle time (FO4 delay) for the various router designs**

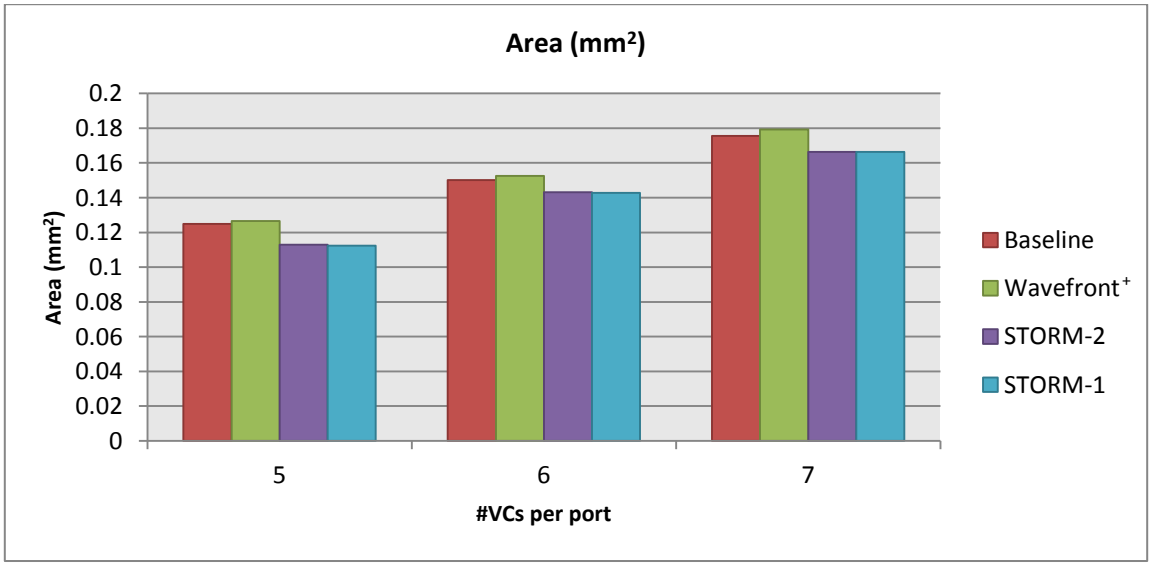


**Figure 6.7: Area ($mm^2$) for the various router designs**

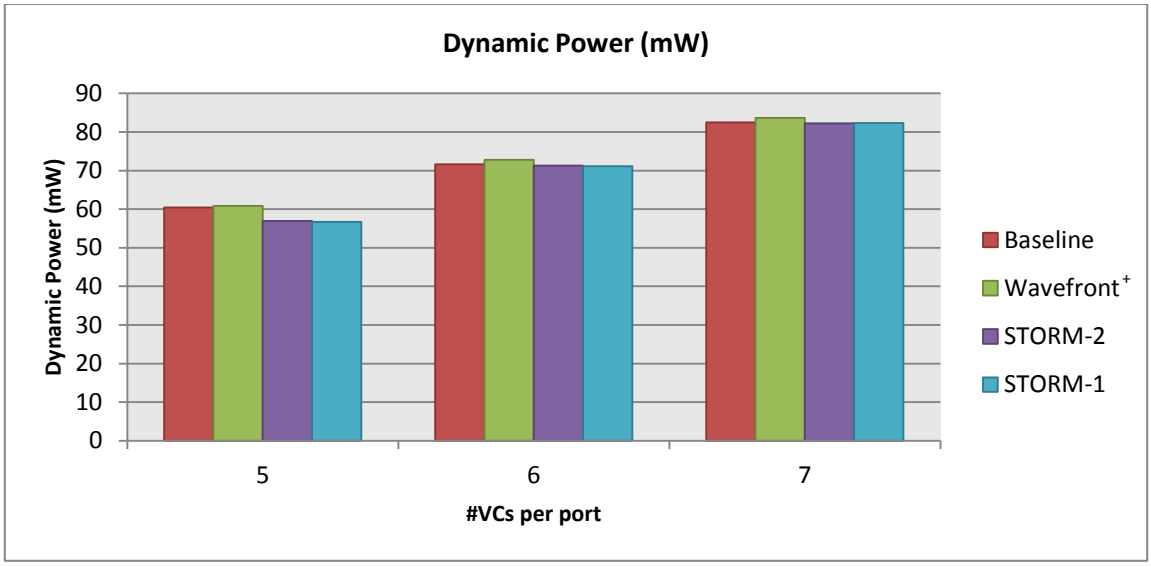


**Figure 6.8: Dynamic power (mW) for the various router designs**

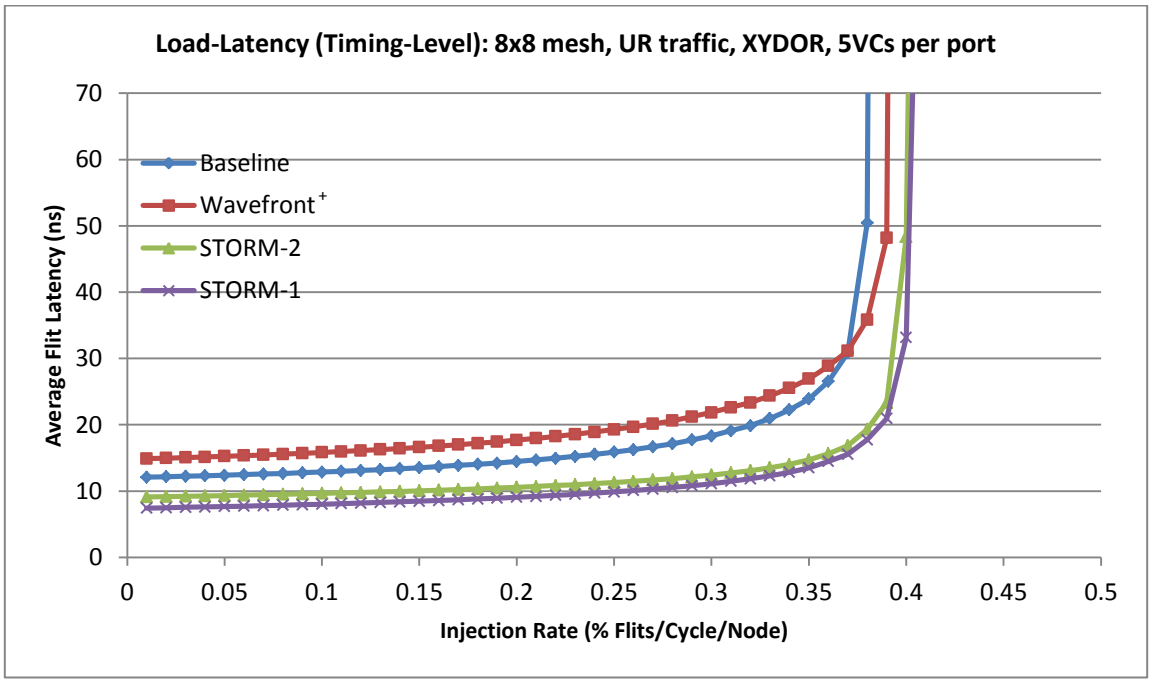


**Figure 6.9: Load-latency curves (timing-level) – 5 VCs per port**

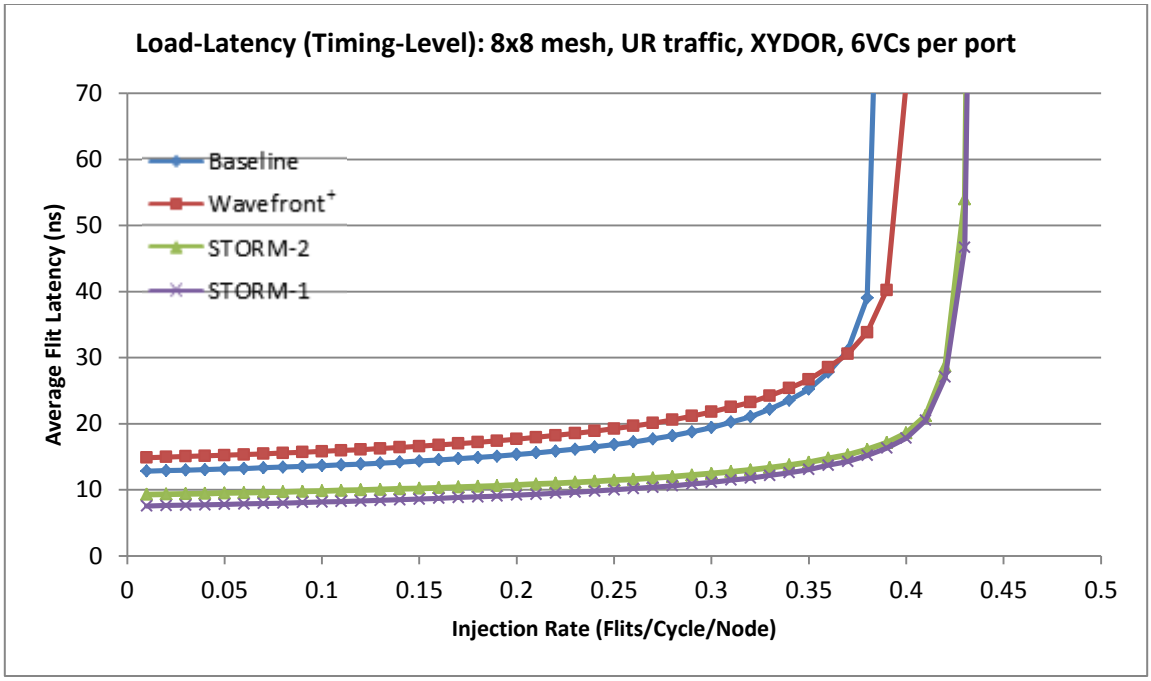


**Figure 6.10: Load-latency curves (timing-level) – 6 VCs per port**

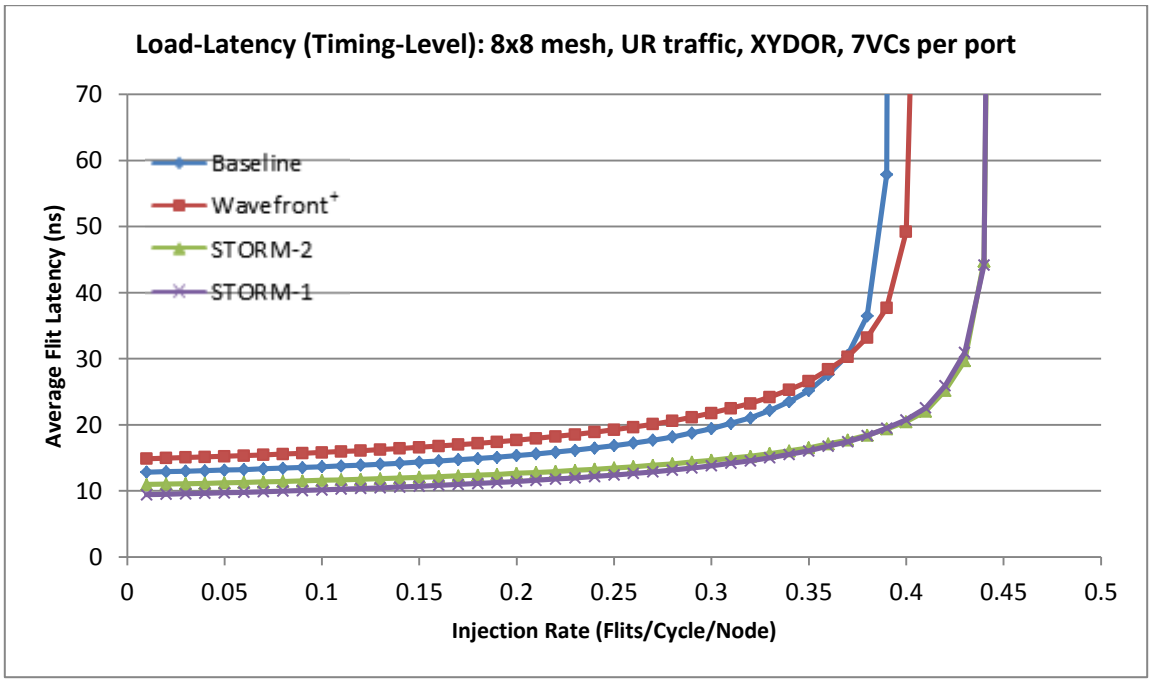


**Figure 6.11: Load-latency curves (timing-level) – 7 VCs per port**

Figures 6.9 to 6.11 show the load-latency curves for uniform random traffic from Figures 6.1 to 6.3, normalized by each design's minimum cycle time (in nanoseconds). STORM-1 improves zero-load latency by up to 41% (for 6 VCs per port) relative to the baseline design, and saturation throughput by up to 14.6% (for 7 VCs per port) relative to the baseline. In addition, STORM occupies lower area than the baseline and wavefront$^{+}$ designs (at 1 GHz) due to the absence of a large monolithic crossbar.

We also synthesize a baseline router by combining the switch allocation and switch traversal stages. For 6 VCs per port, this single-cycle baseline router synthesizes without a timing violation at 1 GHz (1 ns), as opposed to the two-stage baseline router which can be operated at 1.45 GHz (0.69 ns). Combining the pipe stages on a baseline router affects the clock cycle time significantly; simple allocation and switching in STORM enable us to perform stage-merging without paying a high cycle time penalty.

To analyze if a baseline router with fewer than 5 VCs per port can be operated at lower zero-load latency than STORM, we synthesize a baseline router with 1, 2, 3 and 4 VCs per port. Among these configurations, only the 1 VC per port baseline router offers lower zero-load latency on uniform random traffic than a 5 VCs per port STORM-1 (7.1% lower). However, this single VC per port baseline router offers only half the saturation throughput as the 5-VC STORM-1. These results show that despite the lower limit on the number of VCs for STORM, it can still operate at a latency lesser than that of baseline router configurations with fewer VCs.

We also analyze the effect of using the baseline, wavefront$^{+}$, and STORM designs on other synthetic traffic patterns – *transpose* traffic and *bit-complement* traffic.

Figures 6.12 to 6.14 show the time-normalized load-latency curves for transpose traffic, and figures 6.15 to 6.17 show similar curves for bit-complement traffic, for 5, 6 and 7 VCs per port. We see in these cases that while using the STORM designs offers improvement in latency, there is no improvement in overall network throughput relative to the baseline or wavefront$^{+}$ designs. This can be attributed to the absence or near-absence of degrading contention in these traffic patterns. Thus, allocator matching efficiency is not an issue for these traffic patterns, and using different types of allocators does not affect the end result much in terms of throughput. Figure 6.12 shows STORM-2 offering lesser throughput than the other designs – this effect can be attributed to head-of-line blocking caused by restricted VC allocation.

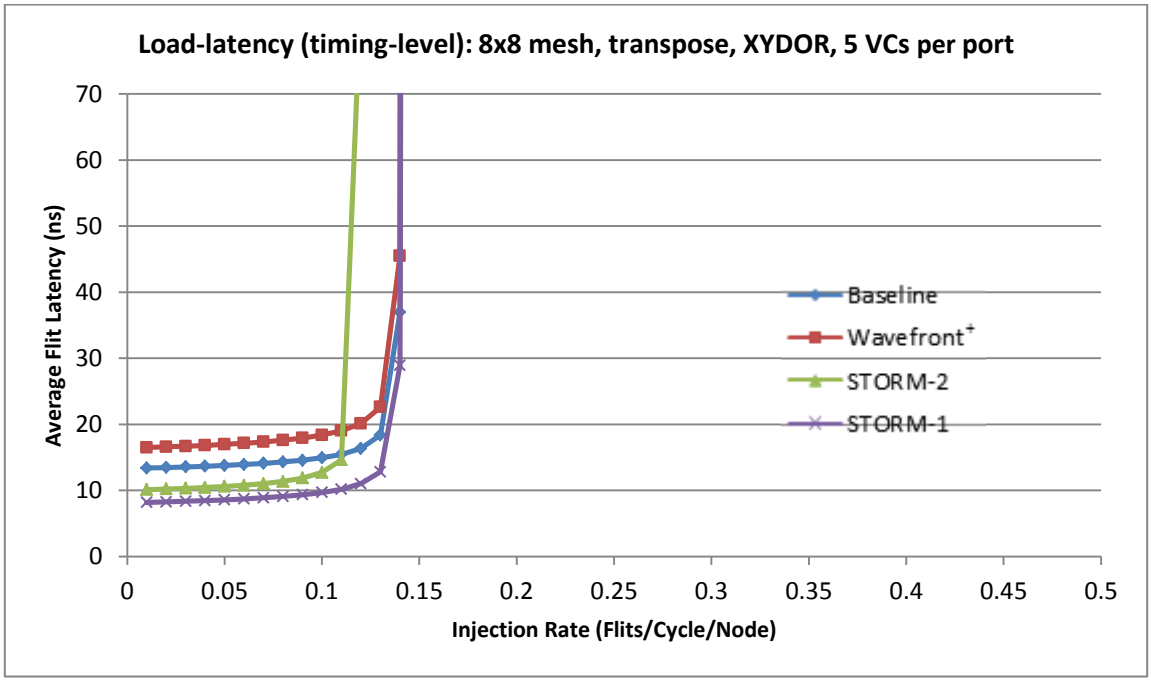


**Figure 6.12: Transpose traffic – Load-latency curves (timing-level) – 5 VCs/port**

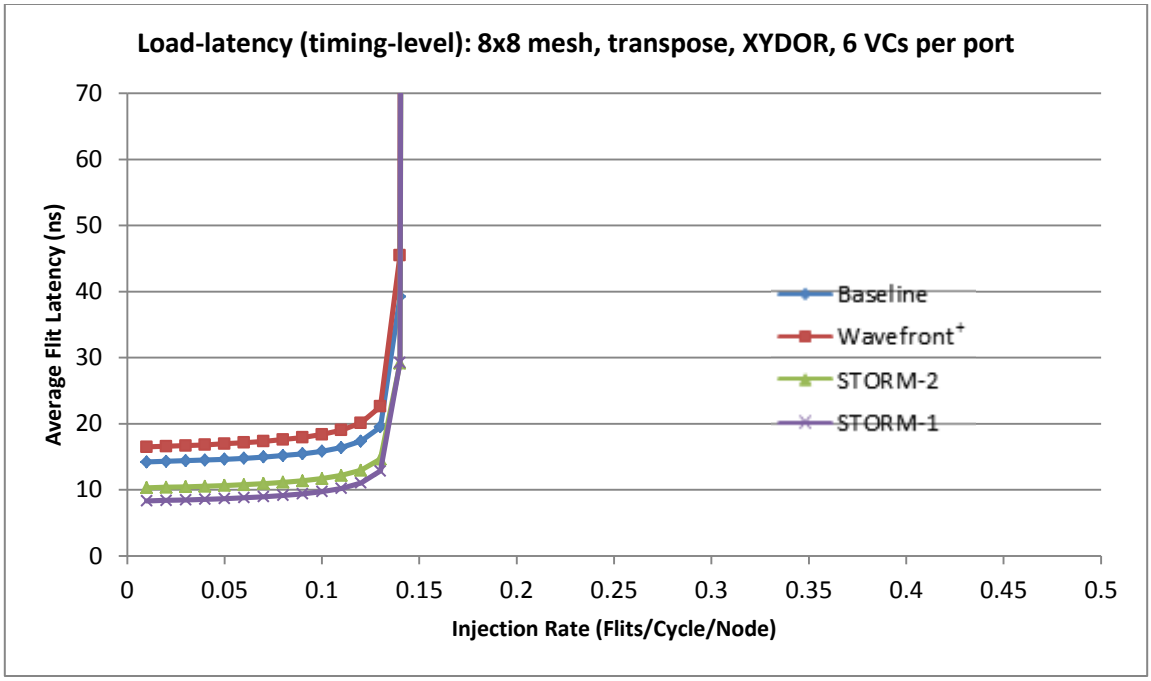


**Figure 6.13: Transpose traffic – Load-latency curves (timing-level) – 6 VCs/port**

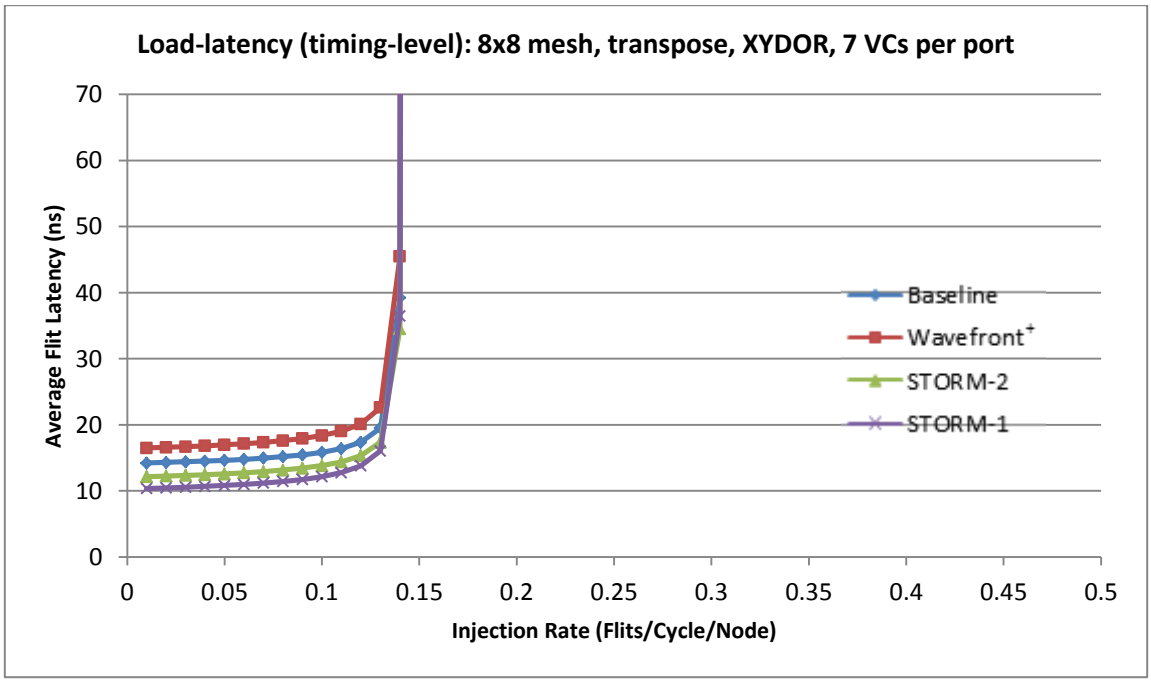


**Figure 6.14: Transpose traffic – Load-latency curves (timing-level) – 7 VCs/port**

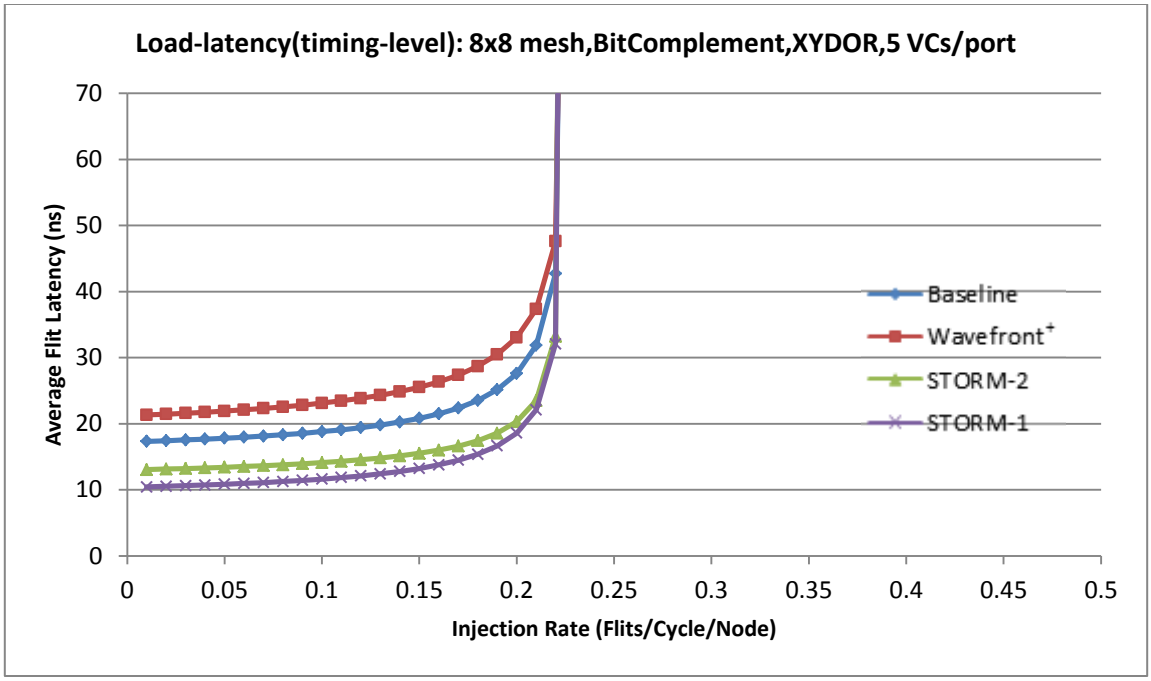


**Figure 6.15: Bit Complement – Load-latency curves (timing-level) – 5 VCs/port**

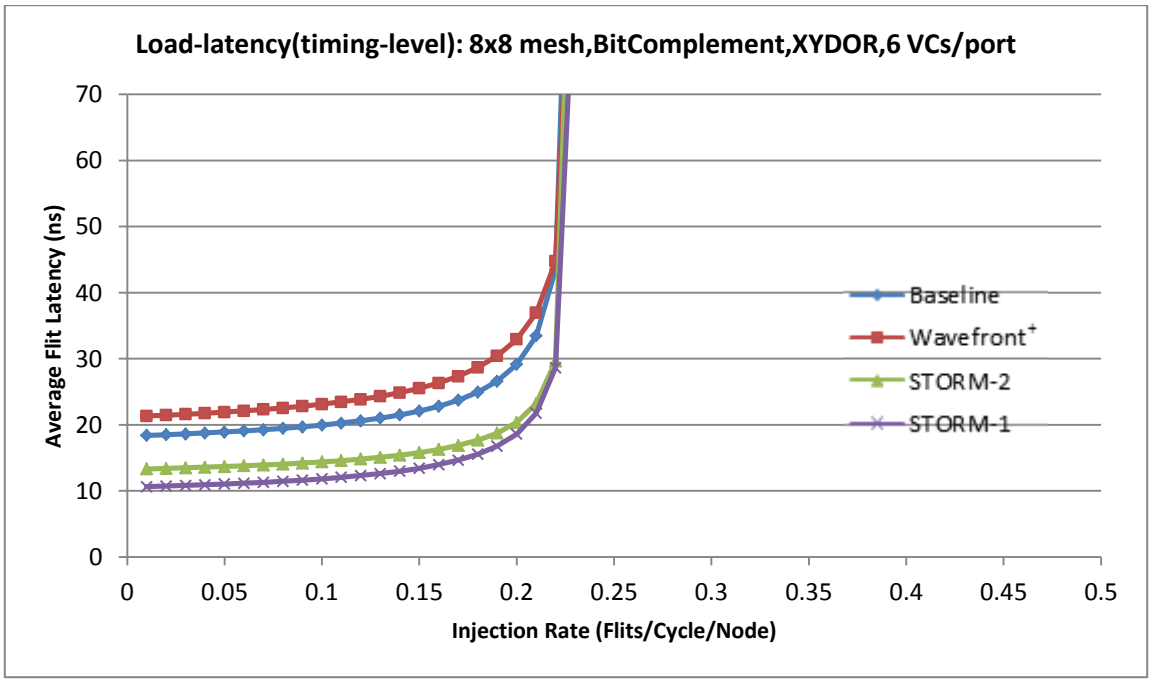


**Figure 6.16: Bit Complement – Load-latency curves (timing-level) – 6 VCs/port**

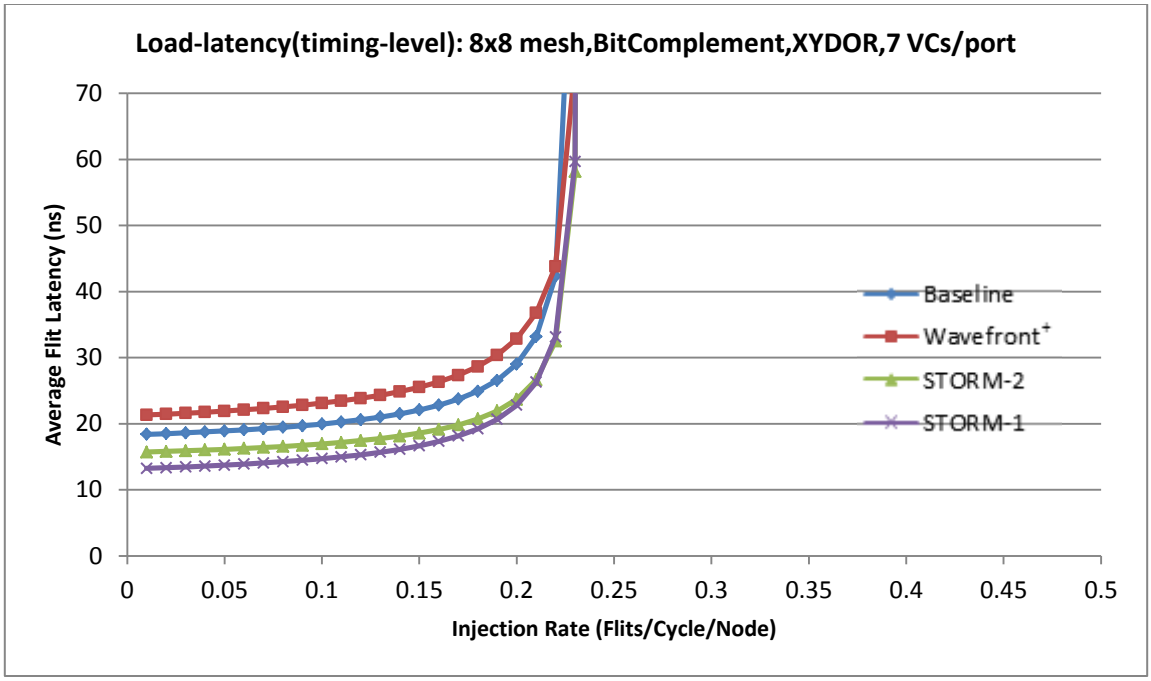


**Figure 6.17: Bit Complement – Load-latency curves (timing-level) – 7 VCs/port**

### 6.3 Application Trace Simulations

While the STORM router design is optimized for uniformly random traffic patterns, it is expected to provide latency and throughput benefits for real applications, too. We analyze the performance of the baseline, wavefront$^{+}$ and the STORM designs on realistic PARSEC [26] workloads using Netrace [25], a trace reader that attempts to emulate actual messages by enforcing dependencies between packets in traces. The PARSEC traces were collected by simulating a 64 core CMP which uses the Alpha ISA. The cores are in-order, with 32KB private L1I and L1D caches employing the MESI coherence protocol, and a 16MB shared L2 cache. For each of the nine benchmarks, a 100 million cycles were simulated in the region of interest.

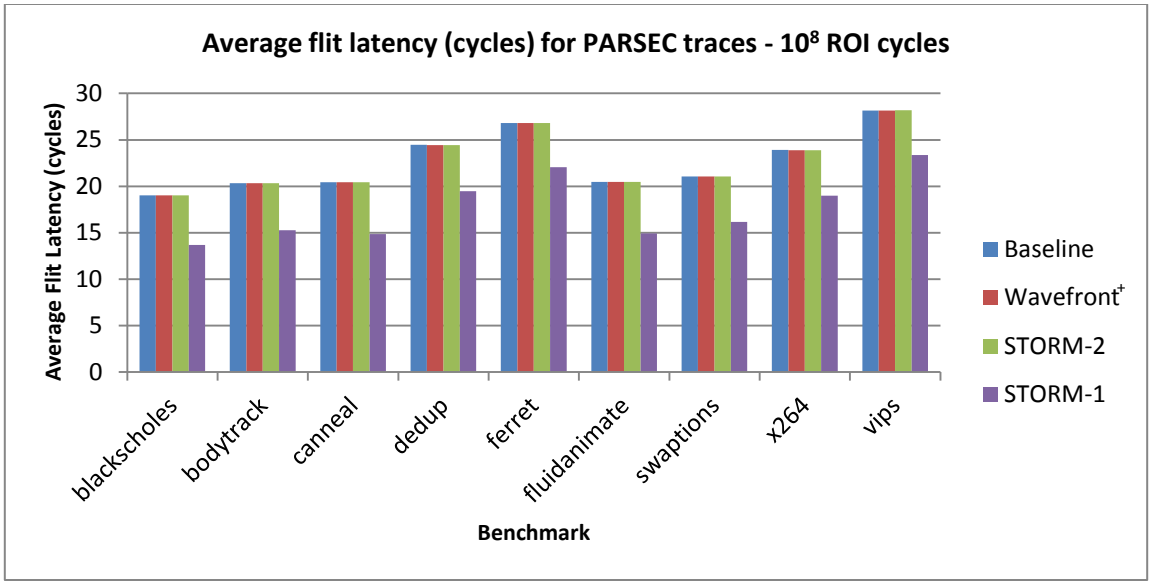


**Figure 6.18: Average flit latency (cycles) for PARSEC traces**

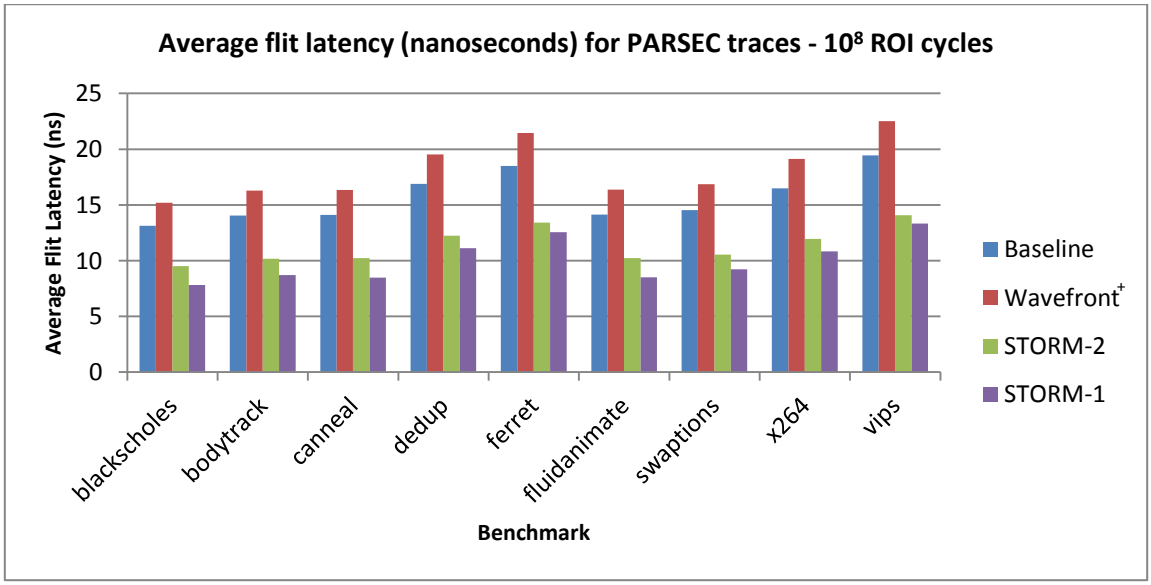


**Figure 6.19: Average flit latency (nanoseconds) for PARSEC traces**

The PARSEC traces were simulated with six 5-flit deep VCs per input port. Figure 6.18 shows the average flit latency for these benchmarks in cycles, and figure

6.19 shows the average flit latency in nanoseconds, for the different router designs. It can be seen from figure 6.19 that STORM offers significant improvement in terms of latency, with STORM-1 being, on average, 36% faster than the baseline router.

We also simulate a 20 million cycle portion (which includes a 5 million cycle warm-up phase) of eight SPLASH2 [27] static benchmark traces on a 7x7 2D mesh, using six 5-flit deep VCs per port for all router designs. The traces are completely static, and the simulator does not enforce packet dependencies. STORM-1 consistently outperforms the baseline and wavefront$^{+}$ designs, offering an average 41% reduction in flit latency. Figures 6.20 and 6.21 show cycle-level and timing-level flit latencies for the SPLASH2 benchmarks while employing the different routers.

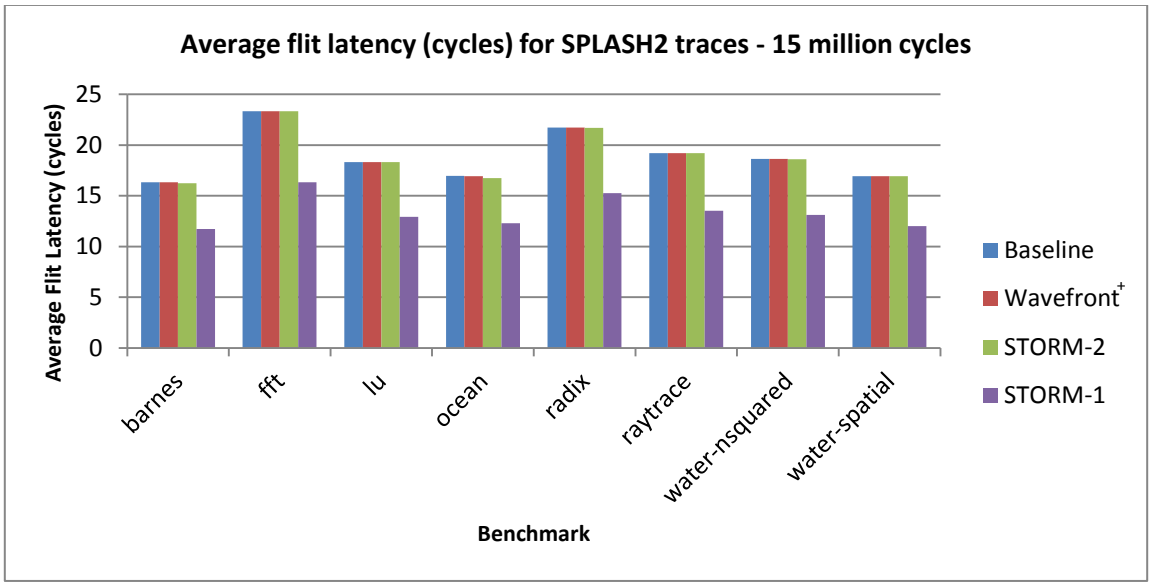


**Figure 6.20: Average flit latency (cycles) for SPLASH2 traces**

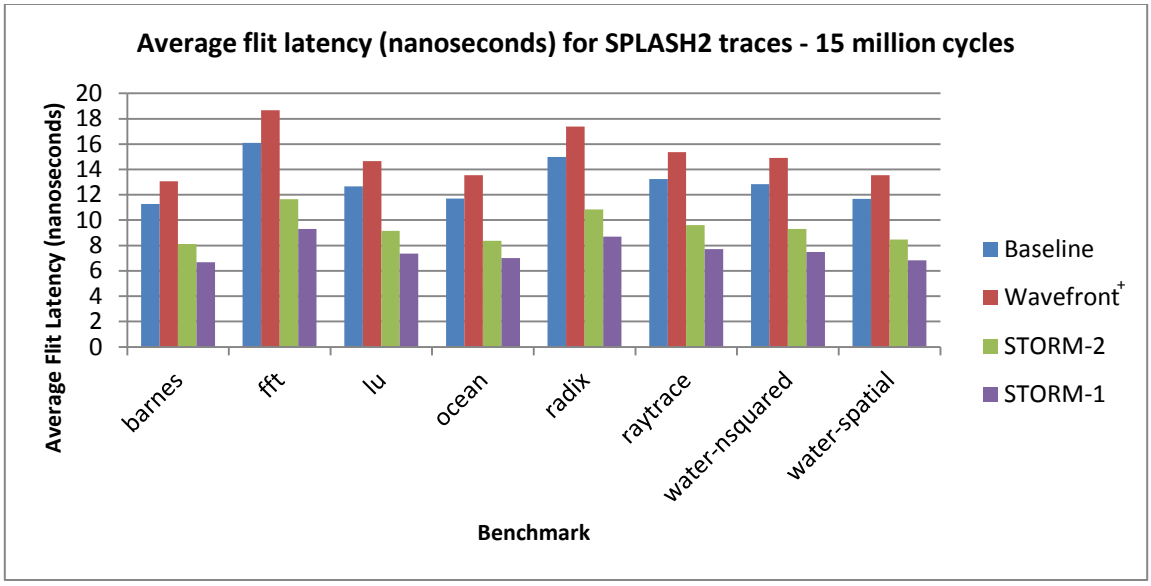


**Figure 6.21: Average flit latency (nanoseconds) for SPLASH2 traces**

# 7. CONCLUSION

We have introduced STORM, a router design that groups virtual channels into path-sets based on router output ports. STORM takes advantage of traffic biases and imbalances in a 2D mesh network employing deterministic, dimension-order routing to improve network throughput compared to the baseline router by removing matching inefficiencies. It also provides significant advantages in terms of latency – a reduction in pipeline depth and circuit complexity, which results in a lower zero-load latency than the baseline router. We also prove that the performance of STORM scales very well with an increasing number of virtual channels. In addition, we show how the proposed design can reduce the network latency for PARSEC and SPLASH2 benchmarks using trace runs on a cycle-accurate simulator. While optimized for uniform-random traffic, STORM exceeds the performance of the baseline and wavefront$^{+}$ routers on both synthetic workloads and on realistic PARSEC and SPLASH2 traces.

# REFERENCES


[1] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Design Paradigm," *IEEE Computer,* vol. 35, no. 1, 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th annual Design Automation Conference*, 2001.

[3] D. U. Becker, "Efficient Microarchitecture for Network-on-Chip Routers," Ph. D. dissertation, Stanford University, 2012.

[4] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," *ACM Computing Surveys,* vol. 38, no. 1, 2006.

[5] W. J. Dally and B. Towles, Principles and Practices of Interconnection Networks, San Francisco, CA: Morgan Kaufmann Publishers Inc., 2004.

[6] W. J. Dally, "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems,* vol. 3, no. 2, 1992.

[7] Y. Tamir and H.-C. Chi, "Symmetric Crossbar Arbiters for VLSI Communication," *IEEE Transactions on Parallel and Distributed Systems,* vol. 4, no. 1, pp. 13-27, 1993.

[8] L. S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, 2001.

[9] A. Mello, L. Tedesco, N. Calazans and F. Moraes, "Virtual Channels in Networks

on Chip: Implementation and Evaluation on Hermes NoC," in *Proceedings of the 18th Annual Symposium on Integrated Circuits and System Design*, 2005.

[10] U. Y. Ogras, J. Hu and R. Marculescu, "Key research problems in NoC design: a holistic perspective," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2005.

[11] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, 2009.

[12] C. A. Nicopoulos, "Network-on-Chip Architectures: A Holistic Design Exploration," Ph. D. dissertation, The Pennsylvania State University, 2007.

[13] R. Mullins, A. West and S. Moore, "Low-Latency Virtual-Channel Routers for On-Chip Networks," in *Proceedings of the 31st annual International Symposium on Computer Architecture*, 2004.

[14] J. Kim, D. Park, T. Theocharides, N. Vijayakrishnan and C. R. Das, "A Low Latency Router Supporting Adaptivity for On-Chip Interconnects," in *Proceedings of the 42nd annual Design Automation Conference*, 2005.

[15] P. Gratz, B. Grot and S. W. Keckler, "Regional Congestion Awareness for Load Balance in Networks-on-Chip," in *Proceedings of the 14th International Symposium on High Performance Computer Architecture*, 2008.

[16] J. Kim, C. Nicopoulos, D. Park, V. Narayanan, M. S. Yousif and C. R. Das, "A

Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks," in *Proceedings of the 33rd annual International Symposium on Computer Architecture*, 2006.

[17] Y. Tamir and G. L. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches," *IEEE Transactions on Computers,* vol. 41, no. 6, 1992.

[18] C. Nicopoulos, D. Park, J. Kim, N. Vijayakrishnan, M. S. Yousif and C. R. Das, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *Proceedings of the 39th annual IEEE/ACM International Symposium on Microarchitecture*, 2006.

[19] A. Kumar, P. Kundu, A. P. Singh, L. S. Peh and N. K. Jha, "A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator in 65nm CMOS," in *Proceedings of the 25th International Conference on Computer Design*, 2007.

[20] J. Kim, "Low-Cost Router Microarchitecture for On-Chip Networks," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009.

[21] R. S. Ramanujam, V. Soteriou, B. Lin and L. S. Peh, "Design of a High-Throughput Distributed Shared-Buffer NoC Router," in *Proceedings of the Fourth ACM/IEEE International Symposium on Networks-on-Chip*, 2010.

[22] Y. Lu, J. McCanny and S. Sezer, "Exploring Virtual-Channel Architecture in FPGA based Networks-on-Chip," in *Proceedings of the IEEE International SoC Conference*, 2011.

[23] B. Zhao, Y. Zhang and J. Yang, "A Speculative Arbiter Design to Enable High-Frequency Many-VC Router in NoCs," in *Proceedings of the Seventh ACM/IEEE International Symposium on Networks-on-Chip*, 2013.

[24] S. Prabhu, B. Grot, P. V. Gratz and J. Hu, "Ocin_tsim-DVFS aware simulator for NoCs," in *Proceedings of the 1st Workshop on SoC Architecture, Accelerators and Workloads*, 2010.

[25] J. Hestness, B. Grot and S. W. Keckler, "Netrace: Dependency-Driven, Trace-Based Network-on-Chip Simulation," in *3rd International Workshop on Network on Chip Architectures (NoCArc)*, 2010.

[26] C. Bienia, S. Kumar, J. P. Singh and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, 2008.

[27] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, New York, 1995.